IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICANT : Motoki Kato et al.
APPLICATION No. : 10/018,838 Group Art Unit : 2621
FILING DATE : June 10, 2002 Examiner :ZHAO, DAQUAN
TITLE : Information Processing Apparatus and Method, Program, and Recorded Medium

Hon. Commissioner of Patents and Trademarks,
Washington, D.C. 20231

SIR:

CERTIFIED TRANSLATION

I, Takashi Narita, am an official translator of the Japanese language into the English language and I hereby certify that the attached comprises an accurate translation into English of Japanese Application No. 2000-183770, filed on April 21, 2000.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

March 13, 2007

Date

Takashi Narita

Takashi Narita

[Document Name] Patent Application

[Reference Number] 0000368703

[Filing Date] April 21, 2000

[To] Hon.Commissioner, Patent Office

[IPC] H04N 5/76

[Inventor]

[Address] c/o Sony Corporation

7-35, Kitashinagawa 6-chome, Shinagawa-ku, Tokyo, Japan

[Name] Motoki Kato

[Inventor]

[Address] c/o Sony Corporation

7-35, Kitashinagawa 6-chome, Shinagawa-ku, Tokyo, Japan

[Name] Toshiya Hamada

[Patent Applicant]

[Identification Number] 000002185

[Name] Sony Corporation

[Representative] Nobuyuki Idei

[Patent Attorney]

[Identification Number] 100082131

[Patent Attorney]

[Name] Yoshio Inamoto

[Telephone Number] 03-3369-6479

[Indication of Charge]

[Number of Prepaid Ledger] 032089

[Amount] 21,000 yen

[List of Document]

[Document] Specification 1

[Document]	Drawing	1
[Document]	Summary	1
[General Power of Attorney Number]		9708842
[Need of Proof]	Yes	

[Name of Document] SPECIFICATION

[Title of the Invention]

Information Processing Apparatus, Inforomation Processing Method and Recording Medium

[Claims]

1. An information processing apparatus comprising:

first recording means for recording AV stream onto or into a recording medium; and

second recording medium for recording, onto or into the recording medium, with the AV stream which has been recorded from a time point when recording of the AV stream has been started by the first recording means up to a time point when such recording has been completed being as one unit, at least one of information relating to recording mode of the AV stream, information relating to average value of recording rate of the AV stream, information relating to a time period where encoded information is the same of the AV stream, information realting to random-accessible position within the AV stream, and information realting to featured image within the AV stream, as attribute information of the AV streams, on the unit basis.

2. The information processing apparatus according to claim 1, further including:

creating means operative so that in the case where user gives an

instruction to continuously reproduce a first AV stream and a second AV stream in the state where the streams of two units or more are recorded on or in the recording medium, the crating means serves to create a third AV stream consisiting of a predetermined portion of the first AV stream and a prodetermined portion of the second AV stream and reproduced when reproduction or playback is switched from the first AV stream to the second AV stream,

wherein the first recording means records, as the AV stream, the third AV stream which has been created by the crating means, and

the second recording means records, as the attribute information, attribute informationn of the third AV stream which has been created by the creating means.

3. An information processing method including:

a first recording control strep of controlling recording of AV stream; and

a second recording control step of recording, with the AV stream in which recording has been controlled from a time point when control of recording of the AV stream has been started at processing of the first recording control step up to a time point when such control of recording has been completed being as one unit, at least one of information realting to recording mode of the AV stream, information relating to average value of

recording rate of the AV stream, information relating to a time period when encoded information is the same of the AV stream, information relating to a random accessible position within the AV stream and inforamtion relating to featured image within the AV stream, as attribute information of corresponding AV stream, on the unit basis.

4. A recording medium where computer readable programm is recorded, the computer readable program including:

a first recording control strep of controlling recording of AV stream: and

a second recording control step of recording, with the AV stream in which recording has been controlled from a time point when control of recording of the AV stream has been started at processing of the first recording control step up to a time point when control of recording has been completed being as one unit, at least one of information realting to recording mode of the AV stream, information realting to average value of recording rate of the AV stream, information relating to a time period where encoded information is the same of the AV stream, information relating to a random accessible position within the AV stream and information relating to featured image within the AV stream, as attribute information of corresponding AV stream,on the unit basis.

[Detailed Description of the Invention]

[0001]

[Technical Field of the Invention]

The present invention relates to an information processing apparatus, an information processing method, and a recording medium. More particularly, the present invention relates to an information processing apparatus, an information processing method, and a recording medium which are adapted for recording, as file, address information of I picture, encoding parameter, change point information and/or information such as mark.

[0002]

[Prior Art]

Recently, a variety of types of optical discs are being proposed as a disc-type recording medium that can be removed from a recording reproducing apparatus. These recordable optical discs have been proposed as a large capacity medium of several GBs and have high anticipation as media for recording AV (Audio Visual) signals. As the sources (supply sources) for digital AV signals to be recorded on the recordable optical disc, there are CS digital satellite broadcasting service and/or BS digital broadcasting services. In addition, ground wave television broadcasting services of the digital system have been also proposed.

[0003]

Here, it is general that digital video signals delivered from these

sources are ordinarily image-compressed in accordance with the MPEG (Moving Picture Experts Group) 2 system. Moreover, in the recording apparatus, recording rate specific to that apparatus is determined. In the case of recording digital video signals through the digital broadcasting service by conventional civilization image storage media, when the analog recording systemm is employed, digital video signal is decoded, and is then recorded after undergone bandwidth limitation. Alternatively, when there is employed digital recording systems including MPEG 1 Video, MPEG 2 Video and DV system, a digital video signal is once decoded, and is then recorded after undergone re-encoding in accordance with the recording rate/encoding system specific to corresponding apparatus.

[0004]

However, in the case of such recording method, since bit stream delivered is once decoded and is then recorded after undergone bandwidth limitation or re-encoding, degradation of picture quantity would take place. In the case of recording image-compressed digital signal, when transmission rate of an inputted digital signal is not above a recording rate of the recording/reproducing apparatus, a method of recording a bit stream delivered as it is without performing decode or re-encode processing results in the fact that degradation of picture quality is the smallest. It should be noted that when transmission rate of image-compressed digital signal is above recording

rate of disc as recording medium, there is a necessity of decoding that digital signal by the recording/reproducing apparatus thereafter to perform re-encoding thereof so that transmission rate becomes equal to an upper limit or less of recording rate of the disc to record it.

[0005]

In addition, in the cse where transmission is performed in accordance the variable rate system in which bit rate of an input digital signal is increased or decreased with elapse of time, since the rotary head has fixed number of rotations, a disc recording apparatus adapted for once storing data into butter to have ability to record it in a burst manner can utilize cpacity of the recording medium without useless as compared to the tape recording system in which the recording rate is fixed rate.

[0006]

As stated above, it is predicted that, in the future where digital broadcasting service becomes main current, there are required recording/reproducing apparatuses adapted for recording digital signals as they are wityout decoding or re-encoding those digital signals, and using disc as recording medium as in the case of dtat streamer.

[0007]

[Problems to be solved by the invention]

In resproducing a recording mediumin on which plural (data of

program consisting of plural data, e.g., video data and/or audio data, etc.) are recorded by apparatuses as described above, it is required to quickly perform processing such as determination of read-out position of AV stream and/or decoding of stream from a recording medium in response to instruction of random access or special reproduction by user. Howevewr, there was the problem that according as quantity of data recorded onto or into a recording medium is increased, it is impossible to quickly perform such a processing.

[0008]

The present invention has been made in view of such circumstances, and its object is to record, as file, address information of I picture, encoded parameter, change point information and/or information such as mark within AV stream to thereby have ability to quickly perform determination of read-out position and/or decode processing of AV stream.

[0009]

[Means for solving the problems]

The information processing apparatus according to claim 1 comprise:

first recording means for recording AV stream onto or into a recording medium; and

second recording medium for recording, onto or into the recording medium, with the AV stream which has been recorded from a time point when recording of the AV stream has been started by the first recording means up to

a time point when such recording has been completed being as one unit, at least one of information relating to recording mode of the AV stream, information relating to average value of recording rate of the AV stream, information relating to a time period where encoded information is the same of the AV stream, information relating to a random-accessible position within the AV stream, and information relating to featured information within the AV stream, as attribute information of the AV streams, on the unit basis.

[0010]

The information processing apparatus further includes: creating means operative so that in the case where user gives an instruction to continuously reproduce a first AV stream and a second AV stream in the state where the streams of two units or more are recorded on or in the recording medium, the creating means serves to create a third AV stream consisting of a predetermined portion of the first AV stream and a predetermined portion of the second AV stream and reproduced when reproduction or platback is switched from the first AV stream to the second AV stream,

wherein the first recording means records, as the AV stream, the third AV stream which has been created by the creating means, and

the second recording means records, as the attribute information, attribute information of the third AV stream which has been created by the creating means.

[0011]

The information processing method according to claim 3 includes; a first recording control strep of controlling recording of AV stream; and a second recording control step of recording, with the AV stream in which recording has been controlled from a time point when control of recording of the AV stream has been started at processing of the first recording control step up to a time point when such control of recording has been completed being as one unit, at least one of information relating to recording mode of the AV stream, information relating to average value of recording rate of the AV stream, information relating to a time period where encoded information is the same of the AV stream, information relating to a random accessible position within the AV stream and information relating to featured image within the AV streams, as attribute information of corresponding AV stream, on the unit basis.

[0012]

The program for the recording medium according to claim 4 includes: a first recording control step of controlling recording AV stream; and a second recording control step of recording, with the AV streamin which recording has been controlled from a time point when control of recording of the AV stream has been started at processing of the first recording control step up to a time point when such control of recording has been completed being as one unit, at

least one of information relating to recording mode of the AV stream, information relating to average value of recording rate of the AV stream, information relating to a time period where encoded information is the same of the AV stream, information relating to a random accessible position within the AV stream and information relating to featured image within the AV streams, as attribute information of corresponding AV stream, on the unit basis.

[0013]

In the information processing apparatus according to claim 1, the information processing method according to claim 3, and the recording medium according to claim 4, with AV stream which has been recorded from the time point when recording of AV stream has been started up to the time point when such recording of AV stream has been completed being as one unit, at least one of information relating to recording mode of AV stream, information relating to average value of recording to rate of AV stream, information relating to the time period where encoded information is the same of the AV stream, information relating to the random accessible position within the AV stream, and information relating to featured image within the AV stream is or are recorded onto or into the recording medium as attribute information of the AV stream on the unit basis.

[0014]

[Best Mode for Carrying out the Invention]

The preferred embodiments of the present invention will now be explained in detail with refrence to the attached drawings. Fig.1 is a view showing an example of the internal configuration of a recording/reproducing apparatus 1 to which the present invention is applied. First, the configuration the portion for performing an operation to record signals input from the external will be explained. The recording/reproducing apparatus 1 is caused to be of the configuration supplied with analog or digital data to have ability to record such data.

[0015]

An analog video signals and an analog audio signal are respectively inputted to terminals 11, 12, respectively. The video signals, which has been inputted to the terminal 11, is outputted to each of an analysis unit 14 and to an AV encoder 15. The audio signal, which has been inputted to the terminal 12, is outputted to the AV encoder 15. The analysis unit 14 extracts feature points such as scene changes, etc., from the inputted video.

[0016]

The AV encoder 15 encodes inputted video and audio signals to output the system information (S) such as encoded video stream (V), encoded audio stream (A) and AV synchronization, etc., to a multiplexer 16.

[0017]

The encoded video stream is a video stream which has been encoded e.g., in accordance with the MPEG (Moving Picture Expert Group) 2 system, and the encoded audio stream is, e.g., an audio stream which has been encoded in accordance with the MPEG1 system, an audio stream which has been encoded in accordance with the Dolby AC3 system. The multiplexer 16 multiplexes the inputted video and audio streams on the basis of inputted system information to output a multiplexed stream to a multiplexed stream analysis unit 18 and to a source packetizer 19 through a switch 17.

[0018]

The multiplexed stream is e.g., an MPEG-2 transport stream or an MPEG2 program stream. The source packetizer 19 encodes the inputted multiplexed stream into an AV stream consisting of source packets in accordance with an application format of a recording medium 100 on which that stream is to be recorded. The AV stream is caused to undergo a predermined processing at an ECC (error correction and coding) unit 20 and a modulation unit 21. The AV stream thus obtained is outputted to a write unit 22, Thus, the write unit 22 writes (records) an AV stream file onto the recording medium 100 on the basis of a control signal outputted from the controller 23.

[0019]

The transport stream of digital television broadcast, etc. which is

inputted from a digital interface or a digital television tuner, is inputted to a terminal 13. There are two recording systems for recording the transport stream inputted to the terminal 13, wherein those streams are a transparent recording system and a system of performing recording after re-encoding is implemented for the purpose of lowering the recording bit rate, etc. The recording system command information is inputted from a terminal 24 as a user interface to a controller 23.

[0020]

In the case of transparently recording an inputted transport stream, the transport stream which has been inputted to the terminal 13 is outputted to the multiplexed stream analysis unit 18 and to the source packetizer 19 through a switch 25. Since processing until AV stream is recorded onto the recording medium 100 at time subsequent thereto is the same processing in the case of encoding and recording the above-described analog inputted audio and video signals, those explanation is omitted.

[0021]

In the case where an inputted transport stream is re-encoded and is subsequently recorded, the transport stream which has been inputted to the terminal 13 is inputted to a demultiplexer 26 through a switch 35. The demultiplexer 26 implements demultiplex processing to the inputted transport stream to extract video stream (V), audio stream (A) and system information

(S).

[0022]

Among the streams (information) which have been extracted by the demultiplexer 26, the video stream is outputted to an AV decoder 27, whilst the audio stream and the system information are outputted to the multiplexer 16. The audio decoder 27 decodes the inputted video stream to output a reproduced video signal to the AV encoder 15. The AV encoder 15 encodes the inputted video signal to output the encoded video stream (V) to the multiplexer 16.

[0023]

On the other hand, the audio stream and the system information, which have been outputted from the demultiplexer 26 and have been inputted to the multiplexer 16, and the video stream, which has been outputted from the AV encoder 15, are multiplexed on the basis of input system information, and are outputted, as a multiplexed stream, to the multiplexed stream analysis unit 18 and to the source packetizer 19 through switch 17. Since processing until an AV stream is recorded onto the recording medium 100 at times subsequent thereto is the same as that in the case of encoding and recording the above-described inputted audio and video signals, their explanation is omitted.

[0024]

The recording/reproducing apparatus 1 of this embodiment records a file of the AV stream onto the recording medium 100, and also records the application database information which explains the file. This application database information is created by control unit 23. The input information to the controller 23 is the feature information for moving picture from the analysis unit 14, the feature information of AV stream from the multiplexed stream analysis unit 18 and the user command or instruction information inputted from the terminal 24.

[0025]

The feature information of the moving picture, which is supplied from the analysis unit 14, is information related to feature image within input moving picture signal, and is, e.g., designation information (mark), such as, for example, program start points, scene change points, CM commercial start and end points, and also includes information of thumbnail image of an image at the designated or specified portion.

[0026]

The feature information of the AV stream from the multiplexed stream analysis unit 18 is the information related to the encoded information of the AV stream to be recorded, and is, e.g., address information of the I-picture in the AV stream, encoding parameters of the AV stream and change point information of the encoding parameters in the AV stream, and information

(mark) related to feature image in video stream.

[0027]

The user designation information from the terminal 24 is the playback or reproduction period designated by the user, character letters for explaining the contents of the playback period, and/or the information such as bookmarks or resuming points which are set at favorite scene by user.

[0028]

On the basis of the aforementioned input information, the controller 23 serves to create a database of the AV stream (Clip), a database of a group (PlayList) of playback periods (PlayItem) of the AV stream, management information (info.drv) of recorded contents of the recording medium 100 (info.dvr) and information of thumbnail images. Similarly to the AV stream, the application database information consisting of these information is processed at the ECC unit 20 and the modulation unit 21, and is then inputted to the write unit 22. The write unit 22 records a database file onto the recording medium 100 on the basis of a control signal outputted from the controller 23.

[0029]

The above-described application database information will be explained later in detail.

[0030]

When the AV stream file (files of video data and audio data) and the application database information recorded on the recording medium 100 in this way are reproduced, the controller 23 first instructs a readout unit 28 to read out the application database information from the recording medium 100. Further, the readout unit 28 reads out the application database information from the recording medium 100. The application database information thus obtained is inputted to the controller 23 after undergone processing by a demodulating unit 29 and an ECC decoder 30.

[0031]

On the basis of application database information, the controller 23 outputs a list of PlayList recorded on the recording medium 100 to a user interface of the terminal 24. The user selects the PlayList, which is desired to be reproduced, from the list of PlayLists. The information relating to PlayList in which reproduction has been designated is inputted to the controller 23. The controller 23 instructs the readout unit 28 to read out the AV stream file necessary for reproducing the PlayList. In accordance with this instruction, the readout unit 28 reads out corresponding AV stream from the recording medium 100 to output the AV stream thus obtained to the demodulating unit 29. The AV stream, which has been inputted to the demodulating unit 29, is caused to undergo a prdetermined processing so that it is decoded. Further, the AV stream thus obtained is outputted through the

processing by the ECC decoder 30 to a source depacketizer 31.

[0032]

The source depacketizer 31 converts the AV stream of the application format, which has been read out from the recording medium 100 and has been caused to undergo a predetermined processing, into a stream which is permitted to be outputted to the demultiplexer 26. The demultiplexer 26 outputs, to the A/V decoder 27, the system information (S) such as the video stream (V), audio stream (A) and the AV synchronization, which constitute the playback period (PlayItem) of the AV stream which has been specified by the controller 23. The AV decoder 27 decodes the video stream and the audio stream to output a reproduction video signal and a reproduction audio signal to each of corresponding terminals 32, 33.

[0033]

Moreover, in the case where information instructing random access reproduction or platback and/or special reproduction or playback is inputted from the terminal 24 as user interface, the controller 23 determines a readout position of the AV stream from the recording medium 100 on the basis of the content of the database (Clip) of the AV stream to instruct the readout unit 28 to read out the AV stream thereof. For example, in the case where the PlayList which has been selected by user is reproduced from a predetermined time point, the controller 23 instracts the readout unit 28 to read out data from

an I-picture having a time stamp closest to the specified time point.

[0034]

Moreover, in the case where fast-forward playback is instructed by user, the controller 23 instructs the readout unit 28 to sequentially read out I-picture data in the AV stream in succession on the basis of database (Clip) of the AV stream.

[0035]

The readout unit 28 reads out data of the AV stream from a specified random access point. The data thus obtained is reproduced after undergone processing by respective units of the succeeding stages.

[0036].

The case where the user edits AV stream recorded on the recording medium 100 will now be explained. In the case where user desired to specify a playback period of the AV stream recorded on the recording medium 100, for example to prepare or create a new playback path, e.g., in the cxase where user desires to create such playback path to reproduce a portion of a singer A from the song program A to subsequently reproduce a portion of the singer A from the song program B, information indicating a start point (IN-point) and an end point (OUT-point) of the playback period is inputted to the controller 23 from the terminal 24 as a user interface. The controller 23 serves to creates a database of the group (PlayList) of playback period

(PlayItem) of the AV streams.

[0037]

In the case where user desires to delete a portion of the AV stream recorded on the recording medium 100, information of the IN-point and the OUT-point of the delete period is inputted to the controller 23. The controller 23 changes the database of the PlayList so as to refer to only the needed AV stream portions. In addition, the controller 23 also instructs the write unit 22 to delete an unnecessary stream portion of the AV stream.

[0038]

The case where the user desires to specify playback period of an AV stream recorded on the recording medium to prepare a new playback path, and desires to interconnect the respective playback periods in a seamless fashion will now explained. In such case, the controller 23 serves to create a database of a group (PlayList) of the playback period (PlayItem) of the AV stream, and to further perform partial re-encoding and re-multiplexing of video stream in the vicinity of the connection point of the playback period.

[0039]

First, the information of picture at the IN-point and information of picture at the OUT-point of a playback period are inputted from the terminal 24 to the controller 23. The controller 23 instructs the readout unit 28 to read out data necessary for reproducing the pictures at the IN-point and at the

OUT-point. Further, the readout unit 28 reads out data from the recording medium 100. The data thus read out is outputted to the demultiplexer 26 through the demodulating unit 29, the ECC decoder 30 and the source packetizer 19.

[0040]

The controller 23 analyzes data which has been inputted to the demultiplexer 26 to determine the re-encoding method for the video stream (change of picture_coding_type and assignment of the quantity of encoding bits for re-encoding) and the re-multiplexing system to deliver the system thus determined to the AV encoder 15 and the multiplexer 16.

[0041]

The demultiplexer 26 then serves to separate the inputted stream into video stream (V), audio stream (A) and system information (S). As the video stream, there are "data to be inputted to the AV decoder 27" and "data to be inputted to the multiplexer 16". The former data is data necessary for performing re-encoding processing, and is decoded by the audio decoder 27. The decoded picture thus obtained is then re-encoded by the AV encoder 15 so that a video stream is provided. The latter data is data copied from an original stream without re-encoding. The audio stream and the system information are directly inputted to the multiplexer 16.

[0042]

The multiplexer 16 multiplexes an input stream on the basis of information which has been inputted from the controller 23 to output a multiplexed stream. The multiplexed stream thus obtained is processed by the ECC unit 20 and the modulation unit 21, and is inputted to the write unit 22. The write unit 22 records an AV stream onto the recording medium 100 on the basis of control signals supplied from the controller 23.

[0043]

The application database information and the operation based on this information such as playback and editing will be explained below. Fig.2 is a view showing the structure of application format. The application format has two layers, of PlayList and Clip, for AV stream management. The Volume Information performs management of all Clips and PlayLists within the disc. Here, pair of one AV stream and associated information thereof is considered as one object, and this is called Clip. The AV stream file is called a Clip AV stream file, and the associated information thereof is called the Clip Information file.

[0044]

For one Clip AV stream file, there is stored data in which MPEG-2 transport stream is arranged so as to take a structure prescribed by the application format. In general, a file is treated as a byte string. The contents of the Clip AV stream file are expanded on the time axis, and entry

points in the Clip is mainly specified by the time basis. When a time stamp of an access point to a predetermined Clip is given, the Clip Information file is useful for finding out address information at which data readout should be started in the Clip AV stream file.

[0045]

PlayList will now be explained with reference to Fig. 3. The PlayList is provided for selecting a playback period that user desires to see from the Clip to have ability to easily edit the playback period thus selected. One PlayList is a set of playback periods in the Clip. One playback period in a predetermined Clip is called PlayItem and is represented by a pair of the IN-point and the OUT-point on the time axis. Accordingly, the PlayList is constitued by a set of plural PlayItems.

[0046]

As the PlayList, there are PlayLists of two types. One type is Real PlayList and the other type is Virtual PlayList. The Real PlayList shares the stream portion of the Clip that it refers. Namely, the Real PlayList takes data capacity corresponding to a stream portion of the Clip that it refers, wherein when Real PlayList is deleted, data of a stream portion of the Clip that it refers is also deleted.

[0047]

The Virtual PlayList does not share Clip data. Accodingly, even if

the Virtual PlayList is changed or deleted, any change does not take place in the contents of the Clip.

[0048]

The editing of the Real Playlist will now be explained. Fig.4(A) is a view showing creation of Real PlayList. In the case where the AV stream is recorded as a new Clip, there results an operation in which the Real PlayList which refers the entire Clip thereof is a newly created.

[0049]

Fig.4(B) is a view relating to the division of the Real PlayList, and this operation is an operation in which the Real PlayList is divided at a desired point so that it is divided into two Real PlayLists. This division operation is performed when in such cases where management of two programs is performed within one clip which is caused to undergo management by a single PlayList, user intends to re-register or re-record programs as separate individual programs. This operation does not lead to change of the Clip contents (division of Clip itself).

[0050]

Fig.4(C) is a view relating to the combining operation of the Real PlayLists. This operation is an operation the operation of combining two Real PlayLists into one new Real PlayList. This combining operation is performed in such cases where, e.g., the user desires to re-register two

programs as a single program. This operation does not lead to change of the Clip (change into the case where there results one Clip by itself).

[0051]

Fig.5(A) is a view relating to deletion of the entire Real PlayList. In the case whre an operation to delete the entirety of a predtermined Real PlayList, the stream portion corresponding thereto of the Clip that the deleted Real PlayList refes is also deleted.

[0052]

Fig.5(B) is a view relating to partial deletion of the Real PlayList. In the case whre a desired portion of the Real PlayList is deleted, the PlayItem corresponding thereto is changed so as to refer only a necessary Clip stream portion. Further, the corresponding stream portion of the Clip is deleted.

[0053]

Fig.5(C) is a view relating to the minimizing of the Real PlayList. This operation is an operation to refer the PlayItem corresponding to the Real PlayList so as to refer only the stream portion of the Clip necessary for Virtual PlayList. The corresponding stream portion of the Clip which is not necessary for the Virtual PlayList is deleted.

[0054]

In the case where the Real PlayList is changed by an operation as described above so that the stream portion of the Clip that the Real PlayList

refers is deleted, there is a possibility that the Virtual PlayList which is using the deleted Clip exists and any problem may take place in the existing Virtual PlayList due to the deleted Clip.

[0055]

In order to prevent that such problem takes place, user is caused, with respect to operation of delete, to display such a message which notifies: "If there exists Virtual PlayList which is referring the stream portion of the Clip that corresponding Real PlayList is refering, and when that Real PlayList is deleted, the Virtual PlayList itself is deleted – is it all right?" to thereby hasten confirmation (alarm) thereafter to execute processing of delete or cancel by user's instruction. Alternatively, the minimizing operation is caused to be performed for the Real PlayList in place of deleting the Virtual PlayList.

[0056]

An operation for the Virtual PlayList will now be explained. Even if an operation is performed for the Virtual PlayList, the contents of the Clip is not changed. Fig.6 is a view relating to the assembling and editing (IN-OUT editing). This operation is an operation of creating PlayItem of the playback period that the user has desired to see to create Virtual PlayList. The seamless connection between PlayItems is supported by the application format (described later).

[0057]

In the case where there exist two Real PlayLists 1, 2 and Clips 1, 2 corresponding to the respective Real PlayLists, user specifies a predetermined time period in the Real PlayList 1 (period from IN1 to OUT1: PlayItem 1) as the playback period, and also specifies, as the period to be reproduced next, a predetermined period in the Real PlayList 2 (period from IN2 to OUT2: PlayItem 2) as the playback period as shown in Fig.6(A), a single Virtual PlayList consisting of PlayItem 1 and the PlayItem2 is created as shown in Fig.6(B).

[0058]

The re-editing of the Virtual PlayList will now be explained. As the re-editing, there are change of IN- or OUT points in the Virtual PlayList, insertion or appending of new PlayItems into the Virtual PlayList, and deletion of PlayItems in the Virtual PlayList. In addition, the Virtual PlayList itself may also be deleted.

[0059]

Fig.7 is a view relating to the audio dubbing (post recording) of audio to the Virtual PlayList. This operation is an operation to register the audio post recording to the Virtual PlayList as a sub path. This audio post recording is supported by the application software. An additional audio stream is added as a sub path to the AV stream of the main path of the Virtual

PlayList.

[0060]

As an operation common to the Real PlayList and the Virtual PlayList, there is an operation of changing (Moving) the playback order of the PlayList as shown in Fig.8. This operation is a change of the playback order of the PlayList in the disc (volume) and is supported by Table Of PlayList (which will be explained later with reference to Fig. 20, etc.), which is defined in the application format. This operation does not lead to change of the Clip content.

[0061]

The mark (Mark) will now be explained. The mark is provided for specifying a highlight or characteristic time in the Clip and in the PlayList. The mark added to the Clip is e.g., a scene change point for specifying a characteristic scene resulting from the content in the AV stream. When the PlayList is reproduced, it may be used by referring the mark of Clip that corresponding the PlayList refers.

[0062]

The mark added to the PlayList is e.g., a bookmark point or a resuming point which is set mainly by user. Setting of the mark to the Clip and to the PlayList is performed by supplementing a time stamp indicating mark time point to the mark list. On the other hand, mark deletion is

removing the time stamp of corresponding mark from the mark list. Accordingly, the AV stream is not changed by any means in accordance with mark setting or by mark deletion.

[0063]

A thumbnail will now be explained. The thumbnail is a still picture added to the Volume, PlayList and Clip. As the thumbnail, there are two kinds of thumbnails. One thumbnail is a thumbnail as a representative picture indicating the content. This is mainly used in a main picture in order to allow user to operate cursor (not shown), etc. to select a thin that he desires to see. The other thumbnail is an image indicating a scene that the mark points out.

[0064]

It is required that the Volume and the respective PlayLists can have representative pictures. It is supposed that the representative picture of the Volume is used in such cases where when disc (recording medium 100 which is assumed to be disc-shaped, and will be referred to as "disc" as occasion may demand) is set at a predetermined portion of the recording/reproducing apparatus 1, still picture representing the content of the disc is first displayed. It is supposed that the representative picture of PlayList is used as still picture for indicating the content of PlayList on the menu picture for selecting PlayList.

[0065]

As the representative picture of the PlayList, it is conceivable that the initial image of the PlayList is employed as the thumbnail (representative picture). However, it is not necessarily limited that the leading picture at the playback time of 0 is an optimum picture in representing the content. In view of the above, the user is permitted to set an arbitrary image as a thumbnail of the PlayList. The above-mentioned two kinds of the thumbnails are called menu thumbnails. Since the menu thumbnails are displayed frequently, it is necessary to read out them at a high speed from the disc. For this reason, it is efficient to store all the menu thumbnails into a single file. It is not necessarily that the menu thumbnail is picture which has been extracted from the moving pictures in the volume, but may be a picture which has been taken thereinto from a personal computer or a digital still camera as shown in Fig.10.

[0066]

On the other hand, it is necessary that plural marks are provided in the clip and the PlayList, whilst it is necessary for recorgnizing the content of the mark position to have ability to easily see image of mark point. The picture indicating such mark point is called a Mark Thumbnail. Accordingly, image which gives basis of the thumbnail is main image from which image of mark point has been extracted rather than a picture which has been taken thereinto

from the external.

[0067]

Fig.11 is a view showing the relation ship between mark attached to the PlayList and the mark thumbnail thereof, whilst Fig.12 is a view showing the relationship between mark attached to the Clip and the mark thumbnail thereof. Since the mark thumbnail is used, in the menu thumbnail, at the time of displaying the detail of the PlayList, it is not requested that the mark thumbnail is read out in a short access time. For this reason, even if every time a thumbnail is required, the recording/reproducing apparatus 1 serves to open a file to read out a portion of that file so that it takes time to some extent, this does not constitute problem.

[0068]

Moreover, for the purpose of decreasing the number of files existing in a volume, it is preferable to store all the mark thumbnails into one file. While the PlayList may one menu thumbnail and plural mark thumbnails, since the user is not required to select the Clip directly (usually, the Clip is selected through PlayList), there is no necessity of providing menu thumbnails.

[0069]

Fig.13 is a view showing the relationship of the menu thumbnails, mark thumbnails, PlayList and Clips in the case where the above-described

fact is taken into consideration. In the menu thumbnail file, there are filed menu thumbnails provided every PlayList. In the menu thumbnail file, there is included a volume thumbnail representing the content of data recorded on the disc. For the menu thumbnail file, there are filed or stored thumbnails created every PlayList and every Clip.

[0070]

The CPI (Characteristic Point Information) will now be explained. The CPI is data included in the Clip information file and is used mainly for finding out a data address where the data readout should be started in the Clip AV stream file when a time stamp of the access point to the Clip is given. In this embodiment, two kinds of CPIs are used, wherein one is EP_map and the other is TU_map.

[0071]

The EP_map is a list of entry point (EP) data, which is extracted from the elementary stream and the transport stream. This has address information finding out the portion of entry point where decode operation should be started of the AV stream. One EP data consists of a pair of presentation time stamp (PTS) and data address in the AV stream of the access unit corresponding to that PTS.

[0072]

The EP_map is used mainly for two purposes. First, EP_map is used

for finding out data address in the AV stream in the access unit referred by the presentation time stamp in the PlayList. Secondly, the EP_map is used for fast forward playback or fast reverse playback. In the case where the recording/reproducing apparatus 1 records the input AV stream when the syntax of the stream can be analyzed, the EP_map is created and is recorded onto the disc.

[0073]

The TU_map has a list of time unit (TU) data which is based on the arrival time of the transport packet inputted through a digital interface. This provides the relationship between the arrival-time-based time and the data address in the AV stream. In the case where the recording and/or reproducing apparatus 1 records an input AV stream, when the syntax of that stream cannot be analyzed, TU_map is created and is recordeded onto the disc.

[0074]

In this embodiment, the self-encoding stream format (SESF) is defined. The SESF is used for encoding analog input signals and for decoding digital input signals (e.g. DV) thereafter to encode the decoded signal into an MPEG-2 transport stream.

[0075]

The SESF defines encoding limit of an elementary stream pertinent

with respect to the MPEG-2 transport stream and the AV stream. When the recording and/or reproducing apparatus 1encodes and records input the SESF stream, EP_map is created and is recorded onto the disc.

[0076]

A digital broadcast stream is recorded onto the recording medium 100 by using either one of systems shown in below. First, the digital broadcast stream is transcoded into an SESF stream. In this case, the recorded stream must becomes in conformity with the SESF. In this case, it is required that EP_map is created prepared and is recorded onto the disc.

[0077]

Alternatively, an elementary stream constituting a digital broadcast stream is transcoded to a new elementary stream and is re-multiplexed with respect to a new transport stream in conformity with the stream format that the organization for standardizing the digital broadcast stream determines. In this case, it is required that EP_map is created and is recorded onto the disc.

[0078]

For example, it is assumed that the input stream is MPEG-2 transport stream in conformity with the ISDB (standard name of digital BS service of Japan), which includes the transport stream containing the HDTV video stream and the MPEG AAC audio stream. The HDTV video stream is transcoded into SDTV video stream to re-multiplex the SDTV video stream

and the original AAC audio stream with respect to TS. The SDTV stream and the transport stream to be recorded must both become in conformity with the ISDB format.

[0079]

As another system of recording digital broadcast stream onto the recording medium 100, there is a susyem to make transparent recording of the input transport stream (to record the input transport stream unchanged), At that time, EP_map is created and is recorded onto the disc.

[0080]

Alternatively, there is a system to transparently record the input transport stream to record input transport stream unchanged. At that time, TU_map is created and recorded onto the disc.

[0081]

The directory and the file will now be explained. The recording and/or reproducing apparatus 1 will be hereinafter described as DVR (Digital Video Recording) as occasion may demands. Fig.14 is a view showing an example of directory structure on the disc. As the directories on the disc of the DVR, there are root directory including "DVR" directory, and the " DVR" directory including "PLAYLIST" directory, "CLIPINF" directory, "M2TS" directory and "DATA" directory as shown in Fig.14. Although directories except for these directories may be created below the root directory,

these directories are assumed to be disregarded in the application format of this embodiment.

[0082]

Below the "DATA" directory, there are stored all files and directories which are prescribed by the DVR application format. The "DVR" directory includes four directories. Below the "PLAYLIST" directory, there are are placed database fields of Real PlayList and Virtual PlayList. This directory may exist even if any PlayList exists only by one.

[0083]

Below "CLIPINF", there is placed database of Clips. This directory also even if any Clip exists only by one. Below "M2TS" directory, there is placed AV stream files. This directory exists even if any AV stream file exists only by one. In the "DATA" directory, there are stored files of data broadcast, such as digital TV broadcast.

[0084]

For the "DVR" directory, files indicated below are stored. "info.dvr" is created below the DVR directory to store the entire information of an application layer. Below the DVR directory, only one info.dvr must exist. The filename is assumed to be fixed to info.dvr. For the "menu.thmb" there is stored the information relating to the menu thumbnails. Below the DVR directory, there must exist 0 or one menu thumbnail. The

filename is assumed to be fixed to "menu.thmb". In the case where any menu thumbnail exists only by one, this file may not exist.

[0085]

For the "mark.thmb" file, there is stored information pertinent to the mark thumbnail image. Below the DVR directory, there must exist 0 or one mark thumbnail. The filename is assumed to be fixed to "menu.thmb". In the case where any menu thumbnail exists only by one, it is sufficient that this file does not exist.

[0086]

For the "PLAYLIST" directory, there is stored two kinds of the PlayList files which are Real PlayList and Virtual PlayList. In "xxxxx.rpls" file, there is stored information relating to one Real PlayList. Respective one files are created for respective Real PlayLists. The filename is "xxxxx.rpls", wherein "xxxxx" denotes five numerical figures from 0 to 9. It is assumed that file extension must be "rpls".

[0087]

For the "yyyyy.vpls", there is stored the information relating to one Virtual PlayList. Respective one files are created every respective Virtual PlayLists. The filename is "yyyyy.vpls" here, "yyyyy" denotes five numerical figures from 0 to 9. It is assumed that file extension must be "vpls".

[0088]

For the "CLIPINF" directory, there are stored respective one files in correspondence with each AV stream files. The "zzzzz.clpi" is a Clip Information file corresponding to one AV stream file (Clip AV stream file or Bridge-Clip stream file). The filename is "zzzzz.clpi", wherein "zzzzz" denotes five numerical figures from 0 to 9. It is auumed that file extension must be "clpi".

[0089]

For the "M2TS" directory, there is stored AV stream file. The "zzzzz.m2ts" file is an AV stream file handled by the DVR system. This is Clip AV stream file or Bridge-Clip AV stream file. The filename is "zzzzz.m2ts", where "zzzzz" denotes five numerical figures from 0 to 9. It is assumed that file extension must be "m2ts".

[0090]

For the "DATA" directory, there is stored data caused to undergo transmission through data broadcasting. As such data, there are, e.g., XML or MPEG files.

[0091]

The syntax and the semantics of each directory (file) will now be explained. First, "info.dvr" directory will be explained. Fig.15 is a view showing the syntax of the "info.dvr" file. The "info.dvr" file is made up

of three objects, which are DVRVoume(), TableOfPlayLists() and MakersPrivateData().

[0092]

The syntax of info.dvr shown in Fig.15 is explained. The TableOfPlayLists_Start_address indicates the leading address of the TableOfPlayLists() with the relative number of bytes from the leading byte of the "info.dvr" file being as unit. The relative number of bytes is counted from 0.

[0093]

The MakersPrivateData_Start_address indicates the leading address of the MakersPrivateData() with the relative number of bytes from the leading byte of the "info.dvr" file being as unit. The relative number of bytes is counted from 0. The padding_word is inserted in association with the syntax of "info.dvr". N1 and N2 are 0 or arbitrary positive integers. Each padding word may assume an arbitrary value.

[0094]

In the DVRVolume(), there is stored the information which describes the contents of the volume (disc). Fig.16 is a view showing the syntax of the DVRVolume(). The syntax of the DVRVolume(), shown in Fig.16 will now be explained. The version_number indicates four character letters indicting the version numbers of the DVRVolume(). The version_number is encoded

into "0045" in accordance with ISO 646.

[0095]

length is represented by 32-bit unsigned integers indicating the number of bytes from immediately after the length field to the trailing end of DVRVolume().

[0096]

In the ResumeVolume(), there is stored the filename of the Real PlayList or the Virtual PlayList reproduced last in the Volume. It should be noted that the playback position when the user has interrupted playback of the Real PlayList or the Virtual PlayList is stored in the resume-mark defined in the PlayListMark().

[0097]

Fig.17 is a view showing the syntax of the ResumeVolume(). The syntax of the ResumeVolume() shown in Fig.17 will be explained. The valid_flag indicates that the resume_PlayList_name field is valid when this 1-bit flag is set to 1, and indicates that it is valid when this flag is et to 0.

[0098]

The 10-byte field of resume_PlayList_name indicates the filename of the Real PlayList or the Virtual PlayList to be resumed.

[0099]

In the UIAppInfoVolume in the syntax of the DVRVolume() shown in

Fig.16, there are stored parameters of the user interface application concerning the Volume. Fig.18 is a view showing the syntax of the UIAppInfoVolume. The semantics thereof will now explained. The 8-bit field of character_set indicates the encoding method for character letters encoded in the Volume_name field. The encoding method corresponds to values shown in Fig.19.

[0100]

The 8-bit field of the name_length indicates the byte length of the Volume name indicated in the Volume_name field. The Volume_name field indicates the name of the Volume. The number of bytes of name_length number from the left of the field indicates valid character letters and indicates the volume name. As values after these valid character letters in the Volume name fiels, any value may be inserted.

[0101]

The Volume_protect_flag is a flag indicating whether or not the contents in the Volume can be shown or pressented to the user without limitations. If this flag is set to 1, the contents of the Volume are permitted to be presented (reproduced) to the user only in case the user has succeeded in correctly inputting the PIN number (password). If this flag is set to 0, the contents of the Volume are permitted to be presented to the user even in case the PIN number is not inputted by the user.

[0102]

At the time point when the user has first inserted a disc into a player, if this flag has been set to 0, or if user has succeeded in correctly inputting the PIN number even in the case where this flag is set to 1, the recording and/or reproducing apparatus 1 demonstrates a list of the PlayList in the disc. The limitations on reproduction of the respective PlayLists are irrelevant to the volume_protect_flag and are indicated by playback_control_flag defined in the UIAppInfo PlayList ().

[0103]

The PIN consists of four numerical figures of from 0 to 9, each of which is encoded in accordance with ISO/IEC 646. The ref_thumbnail_index field indicates the information of a thumbnail image added to the Volume. If the ref_thumbnail_index field is a value except for 0xFFFF, a thumbnail image is added to that Volume. The thumbnail image is stored in menu.thumb file. The image is referred by using the value of the ref_thumbnail_index in the menu.thumb file. If the ref_thumbnail_index field is 0xFFFF, it is indicated that thumbnail image is not added to the Volume.

[0104]

The TableOfPlayList() in the info.dvr syntax shown in Fig.15 will now be explained. In the TableOfPlayList() there is stored the filename of the

PlayList (Real PlayList and Virtual PlayList). All PlayList files recorded in the Volume are included in the TableOfPlayList(). The TableOfPlayList() indicates the playback order of the default of the PlayList in the Volume.

[0105]

Fig.20 is a view showing the syntax of the TableOfPlayList(). The syntax thereof will now be explained. The version_number of the TableOfPlayList() indicates four character letters indicating the version numbers of the TableOfPlayLists. The version_number must be encoded into "0045" in accordance with ISO 646.

[0106]

length is a unsigned 32-bit integer indicating the number of bytes of the TableOfPlayList() from immediately after the length field to the end of the TableOfPlayList(). The 16-bit field of the number_of_PlayLists indicates the number of loops of the for-loop inclusive of the PlayList_file_name. This numerical number must be equal to the number of PlayLists recorded in the Volume. The 10-byte numerical figure of the PlayList_file_name indicates the filename of the PlayList.

[0107]

Fig.21 is a view showing the configuration of another embodiment of the syntax of the TableOfPlayList(). The syntax shown in Fig.21 is caused to be of the configuration inn which UI AppinfoPlayList (which will be

described later) is included into the syntax shown in Fig.20. By employing such configuration including the UIAppInfoPlayList, it becomes possible to create a menu picture only by reading out the TableOfPlayLists. The following explanation will be given on the premise that the syntax shown in Fig.20 is used.

[0108]

The MakersPrivateData in the syntax of the info.dvr shown in Fig.15 will now be explained. The MakersPrivateData is provided to permit the maker of the recording and/or reproducing apparatus 1 to insert private data of the maker in the MakersPrivateData() for special applications of respective makers. The private data of each maker has standardized maker_ID for identifying the maker which has defined it. The MakersPrivateData() may include one or more maker_ID.

[0109]

In the case where a predetermined maker intends to insert private data, the private data of a different maker is already included in the MakersPrivateData(), the new private data is supplemented to the MakersPrivateData() in place of deleting the already existing old private data. Thus, in this embodiment, private data of plural makers are permitted be included in one MakersPrivateData().

[0110]

Fig.22 is a view showing the syntax of the MakersPrivateData. The syntax of the MakersPrivateData shown in Fig.22 will be explained. The version_number thereof indicates four character letters indicating the version numbers of the TableOfPlayLists. The version_number must be encoded into "0045" in accordance with ISO 646. Length is a unsigned 32-bit integer indicating the number of bytes of the MakersPrivateData() from immediately after the length field to the end of the MakersPrivateData().

[0111]

The mpd_blocks_start_address indicates the leading byte address of the first mpd_block() with the number of relative bytes from the leading byte of the MakersPrivateData() being as unit. The number_of_maker_entries is the 16-bit unsigned integer which provides the number of entries of the maker private data included in the MakersPrivateData(). There must not exist two or more maker private data having the same maker_ID values in the MakersPrivateData().

[0112]

The mpd_blocks_size is a 16-bit unsigned integer which provides the size of one mpd_block with 1024 bytes being as unit. For example, if mpd_block_size = 1, it is indicated that the size of one mpd_block is 1024 bytes. The number_of_mpd_blocks is a 16-bit unsigned integer which provides the number of mpd_blocks included in the MakersPrivateData().

The maker_ID is the 16-bit unsigned integer indicating manufacturing maker of the DVR system which has created the maker private data. The value encoded into the maker_ID is specified by the licensor of this DVR format.

[0113]

The maker_model_code is a 16-bit unsigned integer indicating the model number code of the DVR system which has created the maker private data. The value encoded into the maker_model_code is set by the manufacturing maker which has received the license of the format. The start_mpd_block_number is a 16-bit unsigned integer indicating the number of the mpd_block number at which the maker private data is started. The leading end of the maker private data must be aligned with the leading end of the mpd_block. The start_mpd_block_number corresponds to variable j in the for-loop of the mpd_block.

[0114]

The mpd_length is a 32-bit unsigned integer indicating the size of the maker private data on byte basis. The mpd_block is an area in which maker private data is stored. All of the mpd_blocks in the MakersPrivateData() must have the same size.

[0115]

The Real PlayList file and the Virtual PlayList file, in other words, xxxxx.rpls and yyyyy.vpls, will now be explained. Fig.23 is a view showing

the syntax of xxxxx.rpls (Real PlayList) and yyyyy.vpls (Virtual PlayList), which are of the same syntax structure. Each of the xxxxx.rpls and yyyyy.vpls consists of three objects, which are PlayList(), PlayListMark() and MakersPrivateData().

[0116]

The PlayListMark_Start_address indicates the leading address of the PlayListMark() with the number of relative bytes from the leading portion of the PlayList file being as a unit. The relative number of bytes is counted from zero.

[0117]

The MakersPrivateData_Start_address indicates the leading address of the MakersPrivateData() with the relative number of bytes from the leading portion of the PlayList file being as a unit. The number of relative bytes is counted from zero.

[0118]

The padding_word (padding word) is inserted in accordance with the syntax of the PlayList file, wherein N1 and N2 are arbitrary positive integers. Each padding word may take an arbitrary value.

[0119]

PlayList will be further explained although it has been already explained briefly. Each playback term in all Clips except Bridge-Clip (which

will be described later) must be referred by all PlayLists existing in the disc. In addition, two Real PlayLists or more are such that the playback terms shown by their PlayItems must not overlap with each other in the same Clip.

[0120]

Explanation will be further given with reference Fig.24C. For all Clips, there exist corresponding Real PlayLists as shown in Fig.24(A). This rule is observed also after the editing work has been performed as shown in Fig.24(B). Accordingly, all Clips must be viewed by referring any one of Real PlayLists.

[0121]

As shown Fig.24(C), the playback period of the Virtual PlayList must be included in the playback period and in the Bridge-Clip playback term. It is required to prohibit that disc Bridge-Clip which is not referred by any Virtual PlayList exists.

[0122]

The Real PlayList includes the list of the PlayItem, but must not include SubPlayItem. The Virtual PlayList includes the PlayItem list. In the case where the CPI_type shown in the PlayList() is the EP_map type and the PlayList_type is 0 (PlayList including video and audio) , the Virtual PlayList may include one SubPlayItem. In the PlayList() in this embodiment, the SubPlayItem is used only for audio post recording. The number of the

SubPlayItems owned by one Virtual PlayList must be 0 or 1.

[0123]

The PlayList will now be explained. Fig.25 is a view showing the PlayList syntax. The version_number indicates four character letters indicting the version numbers of the PlayList(). It is required that the version_number is encoded into "0045" in accordance with ISO 646. Length is a 32-bit unsigned integer indicating the number of bytes of the PlayList() from immediately after the length field to the end of the PlayList(). The PlayList_type, is an 8-bit field indicating the PlayList type, and one example thereof is shown in Fig. 26.

[0124]

The CPI_type is one-bit flag, which indicates the value of the CPI_type of the Clip referred by the PlayItem() and by the SubPlayItem(). The CPI_types defined in the CPIs of all Clips referred by one PlayList must have the same values. The number_of_PlayItems is a 16-bit field indicating the number of PlayItems existing in the PlayList.

[0125]

The PlayItem_id corresponding to a predetermined PlayItem() is defined by the order in which the PlayItem() appears in the for-loop including the PlayItem(). The PlayItem_id is started from 0. The nimber_of_SubPlayItems is a 16-bit field indicating the number of

SubPlayItems in the PlayList. This value is 0 or 1. An additional audio stream path (audio stream path) is a sort of a sub path.

[0126]

The UIAppInfoPlayList of the PlayList syntax shown in Fig.25 will now be explained. For the UIAppInfoPlayList, there are stored parameters of the user interface application with respect to the PlayList. Fig.27 is a view showing the syntax of the UIAppInfoPlayList. The syntax of UIAppInfoPlayList shown in Fig. 27 will now be explained. The character_set is an 8-bit field, which indicates a method for encoding character letters encoded in the PlayList_name field. The encoding method corresponds to values in conformity with the table shown in Fig.19.

[0127]

The name_length is an 8-bit field, which indicates the byte length of the PlayList name indicated in the PlayList_name field. The PlayList_name field shows the name of the PlayList. The number of bytes of the number of the name_length from the left in the field is valid characters and indicates name of the PlayList. As values after those valid character letters, any value may be inserted.

[0128]

The record_time_and_date is a 56-bit field for storing the date and time where the PlayList was recorded. This field is 14 numerical figures for

year/ month/ day/ hour/minute/ second encoded in binary coded decimal (BCD). For example, 2001/ 12/ 23:01:02:03 is encoded into " 0x20011223010203".

[0129]

The duration is a 24-bit field indicating the total replay time of the PlayList in terms of hour/ minute/ second being as a unit. In this field, six numerical figures are encoded in binary coded decimal (BCD). For example, 01:45:30 is encoded into "0x014530".

[0130]

The valid_period is a 32-bit field indicating the time periods where the PlayList is valid. This field is an area where 8 numerical figures are encoded in 4-bit binary coded decimal (BCD). For example, the recording reproducing apparatus 1 is used such that the PlayList, for which the valid period has lapsed, is automatically erased. For example, 2001/05/07 is encoded into "0x20010507" .

[0131]

The maker_ID is a 16-bit unsigned integer indicating the maker of the DVR player (recording/reproducing apparatus 1) which has updated its PlayList last. The value encoded into maker_ID is assigned to the licensor of the DVR format. The maker_code is a 16-bit unsigned integer indicating the model number of the DVR player which has updated the PlayList last. The

value encoded into the maker_code is determined by the maker which has received the license of the DVR format.

[0132]

In the case where the flag of the playback_control_flag is set to 1, its PlayList is reproduced only when the user can correctly input the PIN number. If this flag is set to 0, the user may view the PlayList without the necessity of inputting the PIN number.

[0133]

In the case where the write_protect_flag is set to 1 as the table is shown in Fig. 28(A), the content of the PlayList is not deleted or changed except the write_protect_flag. If this flag is set to 0, the user can freely delete or change the PlayList. If this flag is set to 1, the recording/reproducing apparatus 1 serves to display such a message to perform re-confirmation before the user deletes, edits or overwrites the PlayList.

[0134]

The Real PlayList, in which the write_protect_flag is set to 0, may exist, the Virtual PlayList, which refers the Clip of the Real PlayList may exist, and the write_protect_flag of the Virtual PlayList may be set to 1. If the user intends to delete the Real PlayList, the recording/reproducing apparatus 1 issues an alarm to the user as to the existence of the aforementioned Virtual

PlayList or "Minimizes" the Real PlayList before erasing the Real PlayList.

[0135]

In the case where is_played_flag is set to 1, as shown in Fig.28B, it is indicated that the PlayList has been reproduced at least once since it has been recorded, whereas, if it is set to 0, it is indicated that the PlayList is not reproduced even once since it has been recorded.

[0136]

archive is a two-bit field indicating whether the PlayList is an original or a copy as shown in Fig.28C. The field of ref_thumbnail_index indicates the information of thumbnail image representative of the PlayList. If the ref_thumbnail_index field is a value which is not 0xFFFF, a thumbnail image representative of the PlayList is added to the PlayList, and the PlayList is stored in the menu.thmb file. That image is referred by using the value of ref_thumbnail_index in the menu.thmb file. If the ref_thumbnail_index field is 0xFFFF, no thumbnail picture representative of the PlayList is added to the PlayList.

[0137]

The PlayItem will now be explained. One PlayItem() elementarily includes the following data: Clip_Information_file_name for specifying the filename of the Clip, pair of IN-time and OUT-time specifying the playback period of Clip; STC_sequence_id that IN-time and OUT-time refer in the case

where the CPI_type defined in PlayList() is EP_map type, and Connection_Condition indicating the connection condition of previous PlayItem and current PlayItem.

[0138]

When PlayList consists of two PlayItems or more, these PlayItems are arrayed in a row, without temporal gap or overlap, on the global time axis of the PlayList. When CPI_type defined in the PlayList is EP_map type and the current PlayList does not have the BridgeSequence(), the pair of IN-time and OUT-time must indicate the same time on the STC continuous domain as that specified by the STC_sequence_id. Such an example is shown in Fig.29.

[0139]

Fig.30 shows the case where when the CPI_type defined in the PlayList() is EP_map type and the current PlayItem has the BridgeSequence(), the rules as now explained are applied. The IN_time of the PlayItem previous to the current PlayItem (shown as IN_time1 in the figure) indicates the time on the STC continuous period specified by the STC_sequence of the previous PlayTime. The OUT_time of the previous PlayItem (shown as OUT_time 1in the figure) indicates the time in Bridge-Clip specified in the BridgeSequenceInfo() of the current PlayItem. This OUT_time must be in conformity with the encoding limitations which will be explained later.

[0140]

The nt PlayItem (shown at IN_time2 in the figure) indicates th specified in the BridgeSequenceInfo() of the current Pla ime also must be in conformity with the encoding li ill be explained later. The OUT_time of PlayItem of tem (shown as OUT_time2 in the figure), indicates th STC continuous period specified by STC_sequen t PlayItem.

[0141]

If the ist() is TU_map type, pair of the IN_time and OUT_time of e the time on the same Clio AV stream, as shown in Fig.3

[0142]

The Pl s as shown in Fig.32. The syntax of the PlayItem sho will be explained. The field of the Clip_informatio dicates the filename of the Clip Information file. The Cli defined by the ClipInfo() of this Clip Information file he Clip AV stream.

[0143]

The STC is an 8-bit field and indicates the STC_sequence_id ontinuous period that the PlayItem refers. If

the CPI_type specified in the PlayList() is TU_map type, this 8-bit field is not significant and is set to 0. IN_time is a 32-bit field and used to store the playback start time of PlayItem. The semantics of IN_time varies depending upon CPI_type defined in the PlayList() as shown in Fig.33.

[0144]

OUT_time is a 32-bit field and is used to store the playback end time of PlayItem. The semantics of the OUT_time varies depending upon CPI_type defined in the PlayList(), as shown in Fig.34.

[0145]

Connection_condition is a 2-bit field indicating the connection condition between the previous PlayItem and the current PlayItem, as shown in Fig.35. Fig.36 is a view showing respective states of Connection_Condition shown in Fig.35.

[0146]

BridgeSequenceInfo will now be explained with reference to Fig.37. This BridgeSequenceInfo is the ancillary information of the current PlayItem and has the following information. Namely, the BridgeSequenceInfo includes Bridge_Clip AV file and a Bridge-Clip_Information_file_name specifying Clip Information file corresponding thereto.

[0147]

Moreover, when corresponding address is an address of a source

packet on the Clip AV stream that the previous PlayItem refers. The first source packet of the Bridge-Clip AV stream is connected subsequently to the source packet. This address is called the RSPN_exit_from_previous_Clip. Further, when corresponding address is also an address of the source packet on the Clip AV stream that the current PlayItem refers. The last source packet of the Bridge_Clip AV stream file is connected before the source packet. This address is called RSPN_enter_to_current_Clip.

[0148]

In Fig.37, RSPN_arrival_time_discontinuity indicates an address of a source packet where a discontinuous point of the arrival time base exists in the Bridge-Clip AV stream file. This address is defined in the ClipInfo().

[0149]

Fig.38 is a view showing the syntax of the BridgeSequenceInfo. The the syntax of BridgeSequenceInfo shown in Fig.38 will be explained. The field of Bridge_Clip_Information_file_name indicates the filename of the Clip Information file corresponding to the Bridge-Clip AV stream file. The Clip_stream_type defined in ClipInfo() of this Clip information file must indicate 'Bridge-Clip AV stream'.

[0150]

The 32-bit field of the RSPN_exit_from_previous_Clip is a relative address of a source packet on the Clip AV stream that the previous PlayItem

refers. The first source packet of the Bridge-Clip AV stream file is connected subsequently to this source packet. The RSPN_exit_from_previous_Clip has a size in which the source packet number is caused to be a unit, and is counted with the value of the offset_SPN defined in the ClipInfo() from the first source packet of the Clip AV stream file that the previous PlayItem refers being as an initial value.

[0151]

The 32-bit field of RSPN_enter_to_curent_Clip is the relative address of the source packet on the Clip AV stream that the current PlayItem refers. The last source packet of the Bridge_Clip_AV stream file is connected before this source packet. The RSPN_enter_to_curent_Clip has a size that is based on the source packet number is caused to be a unit, and is counted with the value of the offset_SPN defined in the ClipInfo() from the first source packet of the Clip AV stream file the current PlayItem refers being as an initial value.

[0152]

The SubPlayItem will now be explained with reference to Fig.39. The use of SubPlayItem() is permitted only if the CPI_type of the PlayList() is the EP_map type. In this embodiment, SubPlayItem is used only for audio post recording. The SubPlayItem() includes the following data. First, it includes Clip_Information_file_name for specifying the Clip that the sub path in the PlayList refers.

[0153]

Moreover, SubPlayItem() also includes SubPath_IN_time and SubPath_OUT_time for specifying the sub path playback period in the Clip. Further, it includes sync_PlayItem_id and start_PTS_of_PlayItem for specifying the time at which reproduction or playback of the sub path is started on the time axis of the main path. The Clip AV stream, which is referred by the sub path, must not contain STC discontinuous points (discontinuous points of the system time base). The clocks of audio samples of the Clip used in the sub path are locked into the clocks of the audio samples of the main path.

[0154]

Fig.40 is a view showing the syntax of the SubPlayItem. The syntax of the SubPlayItem shown in Fig.40 will be explained. The field of the Clip_Information_file_name indicates the filename of the Clip Information file and is used by a sub path in the PlayList. The Clip_stream_type defined in the ClipInfo() of this Clip Information file must indicate the Clip AV stream.

[0155]

An 8-bit field of SubPath_path indicates the sub path type. Here, only '0x00' is set, as shown in Fig.41, while other values are reserved for future use.

[0156]

The 8-bit field of sync_PlayItem_id indicates the PlayItem_id of the PlayItem in which the time at which reproduction or playback of the sub path is started on the time axis of the main path. The value of PlayItem_id corresponding to a predtermined PlayItem is defined in the PlayList() (see Fig.25).

[0157]

A 32-bit field of sync_start_PTS_of_PlayItem indicates the time at which reproduction or playback of the sub path on the time axis of the main path, and indicates the upper 32 bits of the PTS (presentation time stamp) on the PlayItem referenced by the sync_PlayItem_id. For the upper 32 bit field of the SubPath_IN_time, there is stored the playback start time of the sub path. SubPath_IN_time denotes upper 32 bits of the PTS of 33 bit length corresponding to the first presentation unit in the Sub Path.

[0158]

For the upper 32 bit field of subPath_OUT_time, there is stored the playback end time of the Sub path. SubPath_OUT_time indicates upper 32 bits of the value of the Presentation_end_TS calculated by the following equation:

Presentation_end_TS = PTS_OUT + AU_duration

where PTS_out is the PTS of the 33 bit length corresponding to the last

presentation unit of the SubPath, and AU_duration is the 90 kHz based display period of the last presentation unit of the SubPath.

[0159]

PlayListMark() in the syntax of xxxxx.rpls and yyyyy.vpls shown in Fig.23 will now be explained. The mark information relating to the PlayList is stored in this PlayListMark. Fig.42 is a view showing the syntax of PlayListMark. The syntax of the PlayListMark shown in Fig.42 will be explained. The version_number is four character letters indicating the version number of this PlayListMark(). The version_number must be encoded into "0045" in accordance with ISO 646.

[0160]

length is an unsigned 32-bit integer indicating the number of bytes of PlayListMark() from immediately after the length field to the trailing end of the PlayListMark(). The number_of_PlayListMarks is a 16-bit unsigned integer indicating the number of marks stored in the PlayListMark. The number_of_PlayListMarks may be zero. The mark_type is an 8-bit field indicating the mark type and is encoded in accordance with the table shown in Fig.43.

[0161]

For the 32-bit filed of mark_time_stamp, there is stored a time stamp indicating the point at which the mark has been designated. The semantics

of the mark_time_stamp vary depending upon CPI_type defined in the PlayList(), as shown in Fig.44. The PlayItem_id is an 8-bit field specifying the PlayItem where the mark is put. The values of PlayItem_id corresponding to a predetermined PlayItem is defined in the PlayList() (see Fig.25).

[0162]

An 8-bit field of character_set indicates an encoding method for character letters encoded in the mark_name field. The encoding method corresponds to values shown in Fig.19. The 8-bit field of name_length indicates the byte length of the mark name shown in the Mark_name field. The Mark_name field indicates the mark name. The number of bytes of name_length number from the left in this field is the valid character letters and indicates the mark name. As values after those valid character letters in the Mark_name field, any value may be set.

[0163]

The field of the ref_thumbnail_index indicates the information of the thumbnail picture added to the mark. If the field of the ref_thumbnail_index is a value which is not 0xFFFF, a thumbnail picture is added to its mark. That picture image is stored in the mark.thmb file. This picture image is referred in the mark.thmb file by using the value of ref_thumbnail_index which will be explained later. If the ref_thumbnail_index field is 0xFFFF, it is indicated

that no thumbnail picture is added to that mark.

[0164]

The Clip Information file will now be explained. The zzzzz.clpi (Clip Information file) consists of six objects, as shown in Fig.45. They are ClipInfo(), STC_Info(), ProgramInfo(), CPI(), ClipMark() and MakersPrivateData(). For the AV stream (Clip AV stream or Bridge-Clip AV stream) and Clip Information file corresponding thereto, the same string of numerals "zzzzz" is used.

[0165]

The syntax of zzzzz.clpi (Clip Information file) shown in Fig.45 will be explained. The ClipInfo_Start_address indicates the leading address of ClipInfo() with the relative number of bytes from the leading end byte of the zzzzz.clpi file being as a unit. The relative number of bytes is counted from zero.

[0166]

The STC_Info_Start_address indicates the leading address of STC_Info() with the relative number of bytes from the leading end byte of the zzzzz.clpi file as a unit. The relative number of bytes is counted from zero. The ProgramInfo_Start_address indicates the leading address of ProgramInfo() with the relative number of bytes from the leading end byte of the zzzzz.clpi file being as a unit. The relative number of bytes is counted

from 0. The CPI_Start_address indicates the leading address of CPI() with the relative number of bytes from the leading end byte of the zzzzz.clpi file as a unit. The relative number of bytes is counted from zero.

[0167]

The ClipMark_Start_address indicates the leading end address of ClipMark() with the relative number of bytes from the leading end byte of the zzzzz.clpi file as a unit. The relative number of bytes is counted from zero. The_MakersPrivateData Start_address indicates the leading end address of MakersPrivateData() with the relative number of bytes from the leading end byte of the zzzzz.clpi file as a unit. The relative number of bytes is counted from zero. The padding_word is inserted in accordance with the syntax of the zzzzz.clpi file. N1, N2, N3, N4 and N5 must be zero or arbitrary positive integers. The respective padding words may also assume arbitrary values.

[0168]

The ClipInfo will now be explained. Fig.46 is a view showing the syntax of ClipInfo. For the ClipInfo(), there is stored attribute information of AV stream file corresponding thereto (Clip AV stream or Bridge-Clip AV stream file).

[0169]

The syntax of the ClipInfo shown in Fig.46 will be explained. The version_number is the four character letters indicating the version number of

this Clsion_number must be encoded to "0045" in accord46. length is a 32-bit unsigned integer indicating the nu pInfo() from immediately after the length field to the tra lipInfo(). An 8-bit field of Clip_stream_type indicate stream corresponding to the Clip Information file, as show tream of AV streams of respective AV streams will be expla

[0170]

T offset_SPN gives an offset value of the source packet n source packet number of the AV stream (Clip AV stream o AV stream). When the AV stream file is first recorded fset_SPN must be zero.

[0171]

As when the beginning portion of the AV stream file is deleted ffset_SPN may take values except for 0. In this embodime rce packet number (relative address) which refers the offset_S described in the syntax in the form of RSPNxxx (xxx may b., RSPN_EP_start). The relative source packet number has with the source packet number is caused to be a unit and is c first source packet number of the AV stream file with the valu SPN being as the initial value from the first source

packet of the AV stream file.

[0172]

The number of source packets from the first source packet of the AV stream file to the source packet referred by the relative source packet number (SPN_xxx) is calculated by the following equation:

SPN_xxx = RSPN_xxx − offset_SPN.

An example of the case SPN_xxx is 5 is shown in Fig. 49.

[0173]

TS_recording_rate is a 24-bit unsigned integer, and this value gives an input/output bit rate required for the AV stream to the DVR drive (write unit 22) or from the DVR drive (readout unit 28). The record_time_and_date is a 56-bit field for storing the date and time when the AV stream corresponding to the Clip is recorded and is a field in which 14 numerical figures are encoded by 4-bit Binary Coded Decimal (BCD) with respect to year/month/day/hour/minute binary coded decimal (BCD) for 14 numerical figures. For example, 2001/2/23:01:02:03 is encoded into " 0x20011223010203" .

[0174]

The duration is a 24-bit field indicating the total playback time of the Clip in units of hour/minute/second based on arrival time clocks. This field is an area where six numerical figures are encoded by 4-bit binary coded

decimal (BCD). For example, 01:45:30 is encoded into "0x014530" .

[0175]

A flag of time_controlled_flag indicates the recording mode of an AV stream file. If this time_controlled_flag is 1, it is indicated that the recording mode is a mode where recording is performed in such a manner that the file size is proportionate to the time elapsed since recording. This recording mode must satisfy the condition shown by the following equation:

Ts_average_rate*192/188*(t − start_time)− α <= size_clip(t)

<= TS_average_rate*192/188*(t − start_time) + α

where TS_average_rate is a value in which average bit rate of the transport stream of the AV stream file is expressed in units of bytes/second.

[0176]

Moreover, in the above equation, t indicates the time represented in second unit, while start_time is the time point when the first source packet of the AV stream file has been recorded. The size_clip(t) is a value in which size of AV stream file at time t is represented in byte units. For example, in the case where ten Source packets are recorded from Start time to time t, size_clip(t) is 10*192 bytes and α is a constant dependent on TS_average_rate.

[0177]

In the case where time_controlled_flag is set to 0, it is indicated that

the recording mode is a mode where control is not performed such that the time lapse of recording and the file size of the AV stream are proportional to each other. For example, this is the case where input transport stream is recorded in a transparent fashion.

[0178]

In the case where time_controlled_flag is set to 1, the 24-bit field of TS_average_rate indicates the value of TS_average_rate used in the above equation. If time_controlled_flag is set to 0, this field is not significant and must be set to 0. For example, the variable bit rate transport stream is encoded by the following procedure: First, the transport rate is set to the value of TS_recording_rate. Then, the video stream is encoded by the variable bit rate. Further, the transport packet is intermittently encoded resulting from the fact that no null packet is used.

[0179]

The 32-bit field of RSPN_arrival_time_discontinuity is a relative address of a location where arrival timebase discontinuity takes place on the Bridge-Clip AV stream file. The RSPN_arrival_time_discontinuity has a size in which the source packet number is caused to be a unit and is counted with the value of offset_SPN defined in the ClipInfo() as from the first source packet of the Bridge-Clip AV stream file being as initial value. An absolute address in the Bridge-Clip AV stream file is calculated on the base of the

above-mentioned equation:

SPN_xxx = RSPN_xxx − offset_SPN.

[0180]

The 144-bit field of reserver_for_system_use is reserved for system. If is_format_identifier_valid flag is 1, it is indicated that the field of format_identifier is valid. If is_original_network_ID_valid flag is 1, it is indicated that the original_network_ID field is valid. If is_trqnsport_stream_ID_valid flag is 1, it is indicated that the field of is_transport_stream_ID-valid is valid. If is_servece_ID_valid flag is 1, it is indicated that the servece_ID field is valid.

[0181]

When is_country_code_valid flag is 1, it is indicated that the field of country_code is valid. The 32-bit field of format_identifier indicates the value of format_identifier owned that registration descriptor (defined in ISO/IEC13818-1) has in the transport stream. The 16-bit field of transport_stream_ID indicates the value of the transport_stream_ID defined in the transport stream.

[0182]

The 16-bit field of servece_ID indicates the value of servece_ID defined in the transport stream. The 24-bit field of country_code indicates a country code defined by ISO3166. Each character code is encoded by

ISO8859-1. For example, Japan is represented as "JPN" and is encoded into "0x4A 0x50 0x4E". The stream_format_name is 16 character codes of ISO-646 showing the name of a format organization which performs stream definitions of transport streams. A value of '0xFF' is set at invalid by to in this field.

[0183]

format_identifier, original_network_ID, transport_stream_ID, servece_ID, country_code and stream_format_name indicate service providers of transport streams. Thus, it is possible to recognize encoding limitations on audio or video streams and stream definitions of private data streams except for the standard of SI (service information) and/or audio/video streams. These information can be used for judging whether or not the decoder is able to decode the stream, whereby in the case where such decoding is possible, initialization of the decoder system is performed before decoding is started.

[0184]

STC_Info will now be explained. The time period in which STC discontinuous points (discontinuous points of the system time base) is not included in the MPEG_2 transport stream is called the STC_sequence. In the Clip, STC_sequence is specified by the value of STC_sequence_id. Fig. 50 is a view for explining a continuous STC period. The same STC values never appear in the same STC_sequence (it should be noted that the maximum

tine length of Clip is limited as explained below). Accordingly, the same PTS values also never appear in the same STC_sequence. If the AV stream includes N number of STC discontinuous points (N > 0), the Clip system time base is divided into (N+1) number of STC_sequences.

[0185]

For the STC_Info, there is stored address of a location where STC discontinuitie (system timebase discontinuities) takes place. As explained with reference to Fig.51, the RSPN_STC_start indicates the address and the k-th (K>=0) STC_sequence except for the last STC_sequence begins from a time point ot which source packet referred by the (k+1)-th RSPN_STC_start has arrived and ends at the time point when the source packet referred by the k-th RSPN_STC_start has arrived.

[0186]

Fig.52 is a view showing the syntax of the STC_Info. The syntax of STC_Info shown in Fig.52 will be explained. The version_number is four character letters indicating the version number of STC_Info(). The version_number must be encoded into "0045" in accordance with the ISO 646.

[0187]

length is a 32-bit unsigned integer indicating the number of bytes of STC_Info() from immediately after this length field to the end of STC_Info.

If CPI_type of CPI() indicates TU_map type, 0 may be set in this length field. If CPI_type of CPI() indicates EP_map type, the num_of_STC_sequence mut be a value equal to 1 or more.

[0188]

An 8-bit unsigned integer of num_of_STC_sequence indicates the number of sequences in the Clip. This value indicates the number of the for-loops subsequently to this field. The STC_sequence_id corresponding to a predetermined STC_sequence is defined by the order in which the RSPN_STC_start corresponding to that STC_sequence appears in the for-loop including the RSPN_STC_start. The STC_sequence_id is started from 0.

[0189]

The 32-bit field of RSPN_STC_start indicates an address at which the STC_sequence begins on the AV stream file. The RSPN_STC_start indicates an address where discontinuity of system time base takes place in the AV stream file. The RSPN_STC_start may also be a relative address of the source packet having the first PCR of the new system time base in the AV stream. The RSPN_STC_start has a size in which source packet number is caused to be a unit and is counted from the first source packet of the AV stream file with the offset_SPN value defined in ClipInfo() being as an initial value. The absolute address in the AV stream file is calculated by the above-mentioned equation, that is

SPN_xxx = RSPN_xxx - offset_SPN.

[0190]

ProgramInfo in the syntax of zzzz.clip shown in Fig.45 will now be explained with reference to Fig.53. The time period having the following features in the Clip is called program_sequence. Initially the value of PCR_PID is not changed. Moreover, the number of audio elementary streams is not also changed. Further, the PID values in the respective video streams and the encoding information defined by the VideoCodingInfo thereof are not changed. Furthermore, the number of audio elementary streams is not changed. In addition, the PID values of the respective audio streams are not changed, and the encoding information, defined by the AudioCodingInfo thereof, are not changed.

[0191]

Program_sequence has only one system time base at the same time point. Program_sequence has a single PMT at the same time point. For the ProgramInfo(), there is stored an address of a location where the program_sequence begins. RSPN_program_sequence-start indicates that address.

[0192]

Fig.54 is a view showing the syntax of ProgramInfo. The ProgramInfo shown in Fig.54, will be explained. The version_number is

four character letters indicating the version number of ProgramInfo(). The version_number must be encoded into "0045" in accordance with the ISO 646.

[0193]

length is a 32-bit unsigned integer indicating the number of bytes of ProgramInfo() from immediately after this length field to the end of program(info(). If CPI_type of CPI() indicates the TU_map type, this length field may be set to 0. If the CPI_type of CPI() indicates EP_map type, the number_of_programs must be a value equal to 1 or more.

[0194]

An 8-bit unsigned integer of number_of_program_sequences indicates the number of program_sequences in the Clip. This value indicates the number of for-loops subsequent to this field. In the case where program_sequence is not changed in the Clip, 1 must be set in the number of program_sequences. A 32-bit field of RSPN_program_sequence_start is a relative address of a location where the program sequence begins on the AV stream.

[0195]

RSPN_program_sequence_start has a size in which the source packet number is caused to be a unit and is counted with the value of offset_SPN defined in the ClipInfo() from the first source packet of the AV stream file

being as an initial value. The absolute address in the AV stream file is calculated by:

SPN_xxx = RSPN_xxx − offset_SPN.

The values of RSPN_program_sequence_start in the for-loop syntax must appear in the ascending order.

[0196]

A 16-bit field of PCR_PID indicates the PID of the transport packet including valid PCR field in the program_sequence. An 8-bit field of number_of videos indicates the number of for-loops including video_stream_PID and VideoCodingInfo(). An 8-bit field of number_of_audios indicates the number of for loops including audio_stream_PID and AudioCodingINfo(). A 16-bit field of video_stream_PID indicates the PID of the transport packet including valid stream in its program_sequence. VideoCodingInfo() subsequent to this field must explain the content of the video stream referred by its video_stream_PID.

[0197]

A 16-bit field of audio_stream_PID indicates the PID of a transport packet including valid audio stream in its program_sequence. The AudioCodingInfo() subsequent to this field must explain the content of the video stream referenced by its audio_stream_PID.

[0198]

It should be noted that the order in which values of video_stream_PID appear in the for-loop of the syntax must be equal to the order in which PIDs of the video stream are encoded in the PMT valid for the program_sequence. Moreover, the order in which values of audio_stream_PID appear in the for-loop of the syntax must be equal to the order in which PIDs of the audio stream are encoded in the PMT valid for the program_sequence.

[0199]

Fig.55 is a view showing the syntax of VideoCodingInfo in the syntax of the ProgramInfo shown in Fig.54. The syntax of the VideoCoding Info shown in Fig.55 will be explained. An 8-bit field of video_format indicates the video format corresponding to video_stream_PID in ProgramInfo(), as shown in Fig.56.

[0200]

As shown in Fig.57, an 8-bit field of frame_rate indicates the video frame rate corresponding to the video_stream_PID in ProgramInfo(). An 8-bit field of display_aspect_ratio indicates a video display aspect ratio corresponding to video_stream_PID in ProgramInfo() as shown in Fig. 58.

[0201]

Fig.59 is a view showing the syntax of the AudioCodingInfo in the syntax of the ProgramInfo shown in Fig.54. The syntax of the AudioCoding

Info shown in Fig.59 will be explained. An 8-bit field of audio_coding indicates an audio encoding method corresponding to audio_stream_PID in ProgramInfo() as shown in Fig.60.

[0202]

An 8-bit field of audio_component_type indicates an audio component type corresponding to audio_stream_PID in ProgramInfo() as shown in Fig.61. An 8-bit field of sampling_frequency indicates an audio sampling frequency corresponding to audio_stream_PID in ProgramInfo() as shown in Fig.62.

[0203]

The CPI (Characteristics Point Information) in the syntax of zzzzz.clip shown in Fig.45 will now be explained. The CPI is used for allowing the time information in the AV stream and the address in its file to be related with each other. The CPI is of two types, i.e., EP_map and TU_map. As shown in Fig.63, in the case where CPI_type in CPI() is EP_map type, its CPI() includes EP_map. In Fig.64, if CPI_type in CPI() is TU_map, its CPI() contains TU_map. One AV stream has one EP_map or one TU_map. In the case where the AV stream is an SESF transport stream, Clip correspondin thereto must have an EP_map.

[0204]

Fig.65 is a view showing the syntax of CPI. The syntax of CPI shown in Fig.65 will be explained. The version_number is four character

letters indicating the version number of this CPI(). The version_number must be encoded into "0045" in accordance with the ISO 646. Length is a 32-bit unsigned integer indicating the number of bytes from immediately after this length field to the trailing end of the CPI(). The CPI_type is a 1-bit flag and indicates the CPI type of Clip as shown in Fig.66.

[0205]

The EP_map in the CPI syntax shown in Fig.65 will now be explained. There are two types of the EP_map, i.e., EP_map for a video stream and EP_map for audio stream. The EP_map_type in the EP_map differentiates between these EP_map types. If the Clip includes one video streams or more, the EP_map for the video stream must be used. If the Clip does not includes a video stream but includes one audio streams or more, the EP_map for the audio stream must be used.

[0206]

The EP_map for video stream will be explained with reference to Fig.67. The EP_map for the video stream has data of stream_PID, PTS_EP_start and RSPN_EP_start. The stream_PID indicates the PID of the transport packet for performing transmission of a video stream. The PTS_EP_start indicates the PTS of an access unit beginning from the sequence header of the video stream. The RSPN_EP_start indicates the address of a source packet including the first byte of an access unit refereed by

the PTS_EP_start in the AV stream.

[0207]

A sub table, which is called EP_map_for_one_stream_PID(), is created every video stream caused to undergo transmission by transport packet having the same PID. In the case where plural video streams exist in the Clip, the EP_map may include plural EP_map_for_one_stream_PID().

[0208]

The EP_map for audio stream has data of stream_PID, PTS_EP_start and RSPN_EP_start. The stream_PID indicates a PID of transport packet for performing transmission of audio stream. The PTS_EP_start indicates the PTS of access unit of the audio stream. The RSPN_EP_start indicates an address of source packet including the first byte of the access unit referred by PTS_EP_start in the AV stream.

[0209]

The sub table, which is called EP_map_for_one_stream_PID(), is created every audio stream caused to undergo transmission by the transport packet having the same PID. In the case where there exist plural audio streams in the Clip, EP_map may include plural EP_map_for_one_stream_PID().

[0210]

The relationship between EP_map and STC_Info will be explained.

One EP_map_for_one_stream_PID() is created into one table irrespective of discontinuous points of the STC. By making comparison between the value of the RSPN_EP_start and the value of RSPN_STC_start defined in STC_Info(), the boundary of data of EP_map belonging to respective STC_sequences can be understood (see Fig.68). The EP_map must have one EP_map_for_one_stream_PID with respect to continuous stream range caused to undergo transmission by the same PID. As shown in Fig.69, program#1 and program#3 have the same video PID. However, the data range is not continuous, so that EP_map_for_one_stream_PID must be provided for each program.

[0211]

Fig.70 is a view showing the syntax of the system of EP_map. The syntax of the EP_map shown in Fig.70 will be explained. The EP_type is a 4-bit field and indicates the entry point type of the EP_map entry point type, as shown in Fig.71. The EP_type indicates the semantics of data field subsequent to this field. In the case where Clip includes one video stream or more, the EP_type must be set to 0 ('video'). Alternatively, in the case where the Clip does not include no video stream, but includes one audio stream or more, EP_type must be set to 1 ('audio').

[0212]

The 16-bit field of number_of_stream_PIDs indicates the number of

times of loops of the for-loop having number_of_stream_PIDs in the EP_map() as a variable. The 16-bit field of stream_PID(k) indicates the PID of the transport packet for performing transmission of the k-th elementary stream (video or audio stream) referred by EP_map_for_one_stream_PID (num_EP_entries(k)). In the case where EP_type is 0 ('video'), its elementary stream must be video stream. In addition, in the case where EP_type is equal to 1 ('audio'), its elementary stream must be the audio stream.

[0213]

The 16-bit field of num_EP_entries(k) indicates the num_EP_entries(k) referenced by EP_map_for_one_atream_PID, num_EP_entries(k)). The EP_map_for_one_stream_PID_ Start_address(k): This 32-bit field indicates the relative address position at which the EP_map_for_one_stream_ PID(num_EP_entries(k)) begins in the EP_map(). This value is indicated by the size as from the first byte of the EP_map().

[0214]

padding_word must be inserted in accordance with the syntax of EP_map(). X and Y must be zero or arbitrary positive integers. The respective padding words may take arbitrat value.

[0215]

Fig.72 is a view showing the syntax of EP_map_for_one_stream_PID.

The ap_for_one_stream_PID shown in Fig.72 will be expl f the 32-bit field of PTS_EP_start vary depending upon in the EP_map(). In the case where EP_type is equal ield has upper 32 bits of the 33-bit accuracy PTS of the ac with sequence header of the video stream. In the case w qual to 1 ('audio'), this field has upper 32 bits of PTS of e access unit of the audio stream.

[0216]

e 32-bit field of RSPN_EP_start vary depending upon th n EP_map(). In the case where EP_type is equal to 0 (' indicates the relative address of the source packet including he sequence header of the access unit referred by the PTS AV stream. Alternatively, in the case where EP_type i io'), this field indicates the relative address of the source pa first byte of the audio stream of the access unit referred by t in the AV stream.

[0217]

RSP a size in which the source packet number is caused to b counted from the first source packet of the AV stream file, f the offset_SPN, defined in ClipInfo() being as an initial value. address in the AV stream file is calculated by

SPN_xxx = RSPN_xxx - offset_SPN.

It is noted that the value of the RSPN_EP_start in the syntax must appear in the ascending order.

[0218]

The TU_map will now be explained with reference to Fig.73. TU_map creates a time axis on the basis of arrival time clock of source packet (time of the arrive time base). This time axis is called TU_map_time_axis. The origin of TU_map_time_axis is indicated by offset_time in the TU_map(). TU_map_time_axis is divided into predetermined units from offset_time. This unit is called time_unit.

[0219]

In each time_unit in the AV stream, addresses on the AV stream file of the first source packet in the complete form are stored in TU_map. These addresses are called RSPN_time_unit_start. The time at which the k(k >= 0)-th time_unit on the TU_map_time_axis is called TU_start_time(k). This value is calculated on the basis offollowing equation:

TU_start_time(k) = offset_time + k*time_unit_size.

In this case, TU_start_time(k) has accuracy of 45 kHz.

[0220]

Fig.74 shows the syntax of TU_map. The TU_map syntax shown in Fig.74 will be explained. The 32-bit length field of offset_time gives an offset

time with respect to TU_map_time_axis. This value indicates the offset time with respect to the first time_unit in the Clip. The offset_time has a size in which 45 kHz clock derived from the 27 MHz accuracy arrival time clocks being as a unit. In the case where the AV stream is recorded as new Clip, offset_time must be set to 0.

[0221]

The 32-bit field of time_unit_size gives the size of the time_unit, and has a size in which 45 kHz clocks derived from the 27 MHz accuracy arrival time clocks being as a unit. It is preferably that time_unit_size is sett o one second or less (time_unit_size <= 45000). The 32 bit field of number_of_time_unit_entries indicates the number of entries stored in TU_map().

[0222]

The 32-bit field of RSN_time_unit_start indicates the relative address of a location where each time_unit begins in the AV stream. RSPN_time_unit_start has a size in which the source packet number is caused to be a unit and is counted with the value of offset_SPN defined in ClipInfo() from the first source packet of the AV stream file being as an initial value. The absolute address in the AV stream file is calculated by

SPN_xxx = RSPN_xxx - offset_SPN.

It is noted that the value of RSPN_time_unit_start in the for-loop of

the syntax must appear in the ascending order. Ifn the case where there is no source packet in the (k+1)-th time_unit, the (k+1)-th RSPN_time_unit_start must be equal to the number k-th RSPN_time_unit_start.

[0223]

The ClipMark in the syntax of zzzzz.clip shown in Fig.45, will be explained. The the ClipMark is the mark information with respect to clip and is stored in the ClipMark. This mark is not set by a recorder (recording and/or reproducing apparatus 1), but is not set by user.

[0224]

Fig.75 is a view showing the syntax of the ClipMark. The syntax of the ClipMark shown in Fig.75 will be explained. The version_number is four character letters indicating the version number of this ClipMark. The version_number must be encoded into "0045" in accordance with the ISO 646.

[0225]

length is a 32-bit unsigned integer indicating the number of bytes of the ClipMark() as from immediately after the length field to the end of ClipMark(). The number_of_Clip_marks is a 16-bit unsigned integer indicating the number of marks stored in the ClipMark. The number_of_Clip_marks may be equal to 0. Mark_type is an 8-bit field indicating the mark type and is encoded in accordance with the table shown in

Fig.76.

[0226]

mark_time_stamp is a 32-bit field and stores the time stamp indicating a pointer in which a mark has been designated. The semantics of mark_time_stamp varies depending upon CPI_type in the PlayList(), as shown in Fig.77.

[0227]

In the case where CPI_type in CPI() indicates the EP_map type, this 8-bit field of STC_sequence_id indicates the STC_sequence_id of the STC continuous period where marks are placed. In the case where CPI_type in CPI() indicates TU_map type, this 8-bit field has no meaning but is set to 0. The 8-bit field of character_set indicates the an encoding method of character letters encoded in the mark_name field. The encoding method corresponds to the value shown in Fig.19.

[0228]

The 8-bit field of name_length indicates the byte length of the mark name shown in the Mark_name field. This mark_name field indicates the name of mark. The number of bytes of the name_length number from the left in this field is valid character letters and indicates the mark name. As values after those valid character letters in the mark_name field, any value may be inserted.

[0229]

The field of ref_thumbnail_index indicates information of the thumbnail picture added to the mark. In the case where the ref_thumbnail_index field is a value which is not values of 0xFFFF, thumbnail picture is added to its mark, and the thumbnail picture is stored in the mark.thumb file. This picture is referred by using the value of ref_thumbnail_index in the mark.thumb file. In the case where the ref_thumbnail_index field is 0xFFFF, thumbnail picture is not added to its mark.

[0230]

Since MakerPrivateData has been already explained with reference to Fig.22, its explanation will be omitted.

[0231]

Then, thumbnail_information will be explained. A thumbnail picture is stored in a menu.thmb file or in a mark.thmb file. These files are of the same syntax structure and have a single Thumbnail(). For the menu.thmb file, there is stored a menu thumbnail picture, picture representing Volume and picture representing respective PlatyLists. All menu thumbnails are stored into only one menu.thmb file.

[0232]

For the mark.thmb file, there is stored a mark thumbnail picture, i.e., a

picture representing a mark point. All mark thumbnails with respect all PlayLists and Clips are stored in only one mark.thmb file. Since the thumbnails are frequently added or deleted, the operations of addition and partial deletion are able to be easily executed at a high speed. For this reason, Thmbnail() has a block structure. Picture data is divided into several portions each of which is stored in one tn_block. One picture data is stored in successive tn_blocks. In the train of tn_blocks, there may exist tn_block which is not used now. The byte length of one thumbnail picture is variable.

[0233]

Fig.78 is a view showing the syntax of the menu.thmb and the mark.thmb, and Fig.79 is a view showing the syntax of the Thumbnail in the syntax of the menu.thmb and the mark.thmb. The syntax of the Thmbnail syntax shown in Fig79 will be explained. The version_number is the four character letters indicating the version number of this Thumbnail(). The version_number must be encoded into "0045" in accordance with the ISO 646.

[0234]

length is a 32-bit unsigned integer indicating the number of bytes of MakersPrivateData from immediately after the length field to the end of Thumbnail(). tn_block_start_address is a 32-bit unsigned integer indicating the leading byte address of the first tn_block, with the relative number of

bytes from the leading byte of Thumbnail() being as unit. The relative number of bytes is counted from 0. number_of_thumbnails is a 16-bit unsigned integer which provide the number of entries of thumbnail pictures included in the Thumbnail().

[0235]

tn_block_size is a 16-bit unsigned integer which provides one tn_block size with 1024 bytes being as a unit. For example, if tn_block_size = 1, it is indicated that the size of one tn_block is 1024 bytes. number_of_tn_blocks is a 16-bit unsigned integer representing the number of entries of tn_blocks in the Thumbnail(). thumbnail_index is a 16-bit unsigned integer representing the index number of a thumbnail picture represented by thumbnail information corresponding to one for-loop beginning from this thumbnail_index field. The value of 0xFFFF must not be used as thumbnail_index. The thumbnail_index is referred by ref_thumbnail_index in the UIAppInfVolume(), UIAppInfoPlayList(), PlayListMark() and ClipMark().

[0236]

thumbnail_picture_format is an 8-bit unsigned integer representing the picture format of thumbnail picture and takes values shown in Fig.80. "DCF and PNG" in the table are permitted only in "menu_thumb" . The mark thumbnail must take value "0x00" (MPEG-2 Video I-picture).

[0237]

picture_data_size is a 32-bit unsigned integer indicating the byte length of a thumbnail picture on byte basis. start_tn_block_number is a 16-bit unsigned integer indicating tn_block number of the tn_block where data of thumbnail picture begins. The leading portion of the thumbnail picture data must be in correspondence with the leading end of tn_block. The tn_block number begins from 0 and is related to the value of variable k in the for-loop of the tn_block.

[0238]

x_picture_length is a 16-bit unsigned integer representing the number of pixels in a horizontal direction of a frame picture of the thumbnail picture. y_picture_length is a 16-bit unsigned integer representing the number of pixels in the vertical direction of the frame picture of the thumbnail picture. tn_block is an area in which a thumbnail picture is stored. All tn_blocks in Thumbnail() are of the same size (fixed length) and their sizes are defined in size by tn_block_size.

[0239]

Fig. 81 is a view showing, in a model form, how thumbnail picture data are stored in tn_block. As shown in Fig,81, the respective thumbnail picture data begin from the leading portion of tn_block. In the case where picture data is above one tn_block, it is stored by using the successive

subsequent tn_block. By performing storage in this way, management of picture data which has variable size is permitted as fixed length data. Thus, it becomes possible to comply editing such as deletion by simple processing operation.

[0240]

AV stream file will now be explained. The AV stream file is stored in "M2TS" directory (Fig.14). There are two types of the AV stream files, and those streams are Clip AV stream and Bridge-Clip AV stream. Both AV streams must be of the structure of DVR MPEG-2 transport stream file which will be hereinafter defined.

[0241]

First, the DVR MPEG-2 transport stream will now be explained. The structure of DVR MPEG-2 transport stream is as shown in Fig.82. The AV stream file has the structure of a DVR MPEG-2 transport stream. The DVR MPEG 2 transport stream consists of plural Aligned units. The Aligned unit has a size of 6144 bytes (= 2048*3 bytes). The Aligned unit begins from the first byte of the source packet. The source packet has a length of 192 bytes. One source packet consists of TP_extra_header and transport packet. The TP_extra_header has 4 bytes length, and has 188 bytes length.

[0242]

One Aligned unit consists of 32 source packets. The last Aligned

unit in the DVR MPEG-2 transport stream also consists of 32 source packets. Thus, the DVR MPEG-2 transport stream is terminated at a boundary of the Aligned unit. When the number of the transport packets of the input transport stream recorded on the disc is not a multiple of 32, source packet having null packet (transport packet of PID = 0x1FFF) must be used as the last Aligned unit. In the file system, it is not prohibited to add excess information to the DVR MPEG-2 transport stream.

[0243]

The recorder model of the DVR MPEG-2 transport stream is shown in Fig. 83. The recorder shown in Fig.83 is a conceptual model for prescribing a recording process. The DVR MPEG-2 transport stream is in conformity with this model.

[0244]

The input timing of the MPEG-2 transport stream will be explained. The input MPEG-2 transport stream is full transport stream or partial transport stream. The input MPEG-2 transport stream to be inputted must be in accordance with the ISO/IEC13818-1 or ([2]) ISO/IEC13818-9 ([5]). The i-th byte of the MPEG-2 transport stream is inputted at time t(i) simultaneously to both a T-STD (Transport stream system target decoder prescribed by the ISO/IEC 13818-1) 51 and a source packetizer. Rpk is an instantaneous maximum value of the input rate of the transport packet.

[0245]

A 27 MHz PLL52 generates a frequency of 27 MHz clocks. The frequency of 27 MHz clock is locked at a value of PCR (Program Clock Reference) of the MPEG-2 transport stream. An arrival time lock counter 53 is a binary counter serving to count pulses having a frequency of 27 MHz. The arrival_time_clock(i) is a count value of the Arrival time clock counter at time t(i).

[0246]

A source packetizer 54 serves to add TP_extra_header to all transport packets to create a source packet. Arrival_time_stamp indicates times at which the first byte of the transport packet arrives at both the T-STD and the source packetizer. Arrival_time_stamp(k) is a sample value of the Arrival_time_clock(k) as indicating by the following equation:

$$\text{arrival\_time\_stamp}(k) = \text{arrival\_time\_clock}(k) \% 2^{30}$$

where k indicates the first byte of the transport packet.

[0247]

In the case where the time interval of two successively inputted transport packets becomes equal equal to $2^{30}/27000000$ sec (about 40 sec), the difference between the arrival_time_stamps of the two transport packets should be set to $2^{30}/27000000$ sec. The recorder is provided for the case where there results such situation.

[0248]

A smoothing buffer 55 serves to performe smoothing of the bitrate of the input transport stream. The smoothing buffer 55 should not be allowed to overflow. Rmax is an output bitrate of a source packet from the smoothing buffer 55 when the smoothing buffer 55 is not in an empty state. When the smoothing buffer 55 is void in the empty state, the bitrate of output from the smoothing buffer 55 is zero.

[0249]

The parameters of a recorder model of the DVR MPEG-2 transport stream will now be explained. The value of Rmax is given by TS_recording_rate defined in ClipInfo() corresponding to the AV stream file. This value is calculated in accordance with the following equation:

Rmax = TS_recording_rate*192/188

where the value of TS_recording_rate has a size in which bytes/second is caused to be a unit.

[0250]

In the case where the input transport stream is SESF transport stream, Rpk must be equal to TS_recording_rate defined in ClipInfo() corresponding to the AV stream file. In the case where the input transport stream is not SESF transport stream, there may be referred, as this value, values defined in descriptors of the MPEG-2 transport stream, e.g.,

maxi or partial_transport_stream_descriptor.

[0251]

e input transport stream is SESF transport stream, the siz uffer 55 (smoothing buffer size) is zero. In the case w ort stream is not SESF transport stream, values defined of the MPEG-2 transport stream, e.g., smoothi r, short_smoothing_bugger_descriptor or partial_ scriptor, etc. may be referred as the size of the smoothi

[0252]

A corder) and a reproducing unit (player) are each required a sufficient size. The default buffer size is 1536 bytes.

[0253]

Th the DVR MPEG-2 transport stream will now be explained. v showing the player model of the DVR MPEG-2 transport st a conceptual model for prescribing the replay or reproductio DVR MPEG-2 transport stream is in conformity with this mo

[0254]

A 27 generates a frequency of 27MHz. The error range

of the 27 MHz frequency must be ± 30 ppm (27000000±810 Hz). An arrival time clock counter 62 is a binary counter serving to count pulses having a frequency of 27 MHz. Arrival_time_clock(i) is a count value of the Arrival time clock counter at time t(i).

[0255]

In the smoothing buffer 64, Rmax is an input bitrate of a source packet to the smoothing buffer when the smoothing buffer is not in full state. When the smoothing buffer is in full state, the input bitrate to the smoothing buffer is zero.

[0256]

The output timing of the MPEG-2 transport stream will be explained. When arrival_time_stamp of the current source packet is equal to the value of the 30 LSB bits of arrival_time_clock(i), the transport packet of the source packet is extracted from the smoothing buffer. Ppk is an instantaneous maximum value of the transport packet rate. The smoothing buffer should not be allowed to overflow.

[0257]

The parameters of the player model of the DVR MPEG-2 transport stream are the same as the parameters of the recorder model of the above-described DVR MPEG-2 transport stream.

[0258]

Fig.85 is a view showing the syntax of the Source packet. transport_packet() is an MPEG-2 transport packet prescribed in ISO/IEC 13818-1. The syntax of the TP_Extra_header shown in Fig.86 will now be explained. copy_permission_indicator is an integer representing copy limitations of the payload of the transport packet. The copy limitations may be set to copy free, no more copy, copy once or copy prohibited. Fig.87 shows the relation ship between values of copy_permission_indicator and the modes designated by these values.

[0259]

copy_permission_indicator is added to all transport packets. In the case where an input transport stream is recorded by using the IEEE1394 digital interface, the value of copy_permission_indicator may be related to the value of EMI (Encryption Mode Indicator) in the IEEE1394 isochronous packet header. In the case where input transport stream is recorded without using the IEEE1394 digital interface, the value of copy_permission_indicator may be related to the value of the CCI embedded in the transport packet. In the case of performing self-encoding of an analog signal input, the value of the copy_permission_indicator may be related to the value of CGMS-A of analog signal.

[0260]

Arrival_time_stamp is an integer value having a value specified by

arrival_time_stamp in the following equation.

arrival_time_stamp(k) = arrival_time_clock(k)%230.

[0261]

In order to define Clip AV stream, the Clip AV stream must have the structure of the DVR MPEG-2 transport stream in which definition as described above is made. arrival_time_clock(i) must increase continuously in the Clip AV stream. Even if a discontinuous point of the system time base (STC base) exists in the Clip AV stream, the arrival_time_clock(i) of that Clip AV stream must increase continuously.

[0262]

The maximum value of the difference of the arrival_time_clock(i) between the beginning and the end in the Clip AV stream must be 26 hours. This limitation guarantees that in the case where there exists no discontinuous point of the system time base (STC base) in the MPEG-2 transport stream, PTS (Presentation Time Stamp) of the same value never appears in the Clip AV stream. The MPEG2 system standards prescribes that the wraparound period of PTS should be 233/90000 sec (about 26.5 hours).

[0263]

In order to define the Bridge-Clip AV stream, the Bridge-Clip AV stream must have the structure of the DVR MPEG-2 transport stream defined as described above. The Bridge-Clip AV stream must include discontinuous

point of one arrival time base. The transport stream before and after the discontinuous point of the arrival time base must be in conformity with the limitation of encoding which will be explained later and also must be in conformity with the DVR-STD which will be explained later.

[0264]

In this embodiment, seamless connection of the video and audio between PlayItems in editing is supported. This seamless connection between the PlayItems guarantees the player/recorder to realize the " continuous data supply" and "seamless decoding". The "continuous supply of data" means that the file system can guarantee the decoder to supply data at a necessary bitrate in order that underflow of buffer does not take place. In order to guarantee real-time characteristic of data to have ability to read out data from the disc, tata are stored in units of successive blocks having sufficient size.

[0265]

The " seamless decoding" means that the player can display audio/video data recorded on the disc without producing pause or gap in reproduction output of the decoder.

[0266]

The AV stream that the seamlessly connected PlayItem refers will be explained. Whether or not the connection between the previous PlayItem

and the current PlayItem is guaranteed so as to permit seamless display can be judged from the connection_condition field defined in the current PlayItem. As a method for seamless connection between PlayItems, there are a method using Bridge-Clip and a method without using Bridge-Clip.

[0267]

Fig.88 shows the relation ship between the previous PlayItem and the current PlayItem in the case where Bridge-Clip is used. In Fig. 88, stream data that the player reads out is indicated in the shaded state. The TSI shown in Fig. 88 consists of shaded stream data of Clip1 (Clip AV stream) and shaded stream data previous to RSPN_arrival_time_discontinuity of Bridge-Clip.

[0268]

The shaded stream data of Clip1 of TSI is stream data from an address of a stream necessary for decoding a presentation unit corresponding to IN_time of the previous PlayItem (shown at IN_time in Fig.88) up to a source packet referred by RSPN_exit_from_previous_Clip. The shaded stream data previous to RSPN_arrival_time_discontinuity of Bridge-Clip included in TSI is stream data from the first source packet of Bridge-Clip to a source packet immediately before source packets referred by RSPN_arrival_time_discontinuity.

[0269]

On the other hand, TS2 in Fig.88 consists of shaded stream data of Clip2 (Clip AV stream) and shaded stream data subsequent to RSPN_arrival_time_discontinuity of Bridge-Clip. The shaded stream data subsequent to RSPN_arrival_time_discontinuity of Bridge-Clip included in TS2 is stream data from the source packet referred by RSPN_arrival_time_discontinutity up to the last source packet of Bridge_Clip. The stored stream data of Clip 2 of TS2 us stream data from source packet referred by RSPN_enter_to_current_Clip to an address of a stream necessary for decoding the presentation unit corresponding to OUT_time of the current PlayItem (shown at OUT_time2 in Fig.88).

[0270]

Fig.89 shows the relationship between the previous PlayItem and the current PlayItem in the case where Bridge-Clip is not used. In this case, the stream data that the player reads out is shown in the shaded state. The TSI in Fig. 89 consists of shared stream data of Clip 1 (Cli AV stream). The shared stream data of TS1 is data beginning from the address of a stream necessary for decoding the presentation unit corresponding to IN_time of previous PlayItem (shown at IN_time1 in Fig.89) and up to the last source packet of Clip1. On the other hand, TS2 in Fig.89 consists of shaded stream data of Clip2 (Clip AV stream).

[0271]

The shaded stream data of Clip2 of TS2 is stream data beginning from a first source packet of Clip2 and up to an address of a stream necessary for decoding the presentation unit corresponding to the OUT_time of the current PlayItem (shown at OUT_time2 in Fig.89).

[0272]

In Figs.88 and 89, TS1 and TS2 are continuous streams of source packets. Let now consider stream prescriptions of TS1 and TS2 and connection conditions therebetween. As a limitation of the encoding structure of the transport stream, the number of video streams included in TS1 and TS2 must be 1. The number of audio streams included in TS1 and TS2 must be 2 or less. The numbers of audio streams included in TS1 and TS2 must be equal to each other. It is noted that an elementary stream or a private stream except for the above may be included in TS1 and/or TS2.

[0273]

The limitation of video bitstream will now be explained. Fig.90 is a view showing an instance of seamless connection shown in the picture display sequence. In order that video stream can be displayed seamlessly at connectyion point, unnecessary pictures displayed after OUT_time (OUT_time of Clip1) and before the IN_time2 (IN_time of Clip2) must be removed by a process of re-encoding a partial stream of the Clip in the vicinity of the connection point.

[0274]

An example of realizing seamless connection by using BridgeSequence in such cases as shown in Fig.90 is shown in Fig. 91. The video stream of Bridge-Clip previous to RSPN_arrival_time_discontinuity consists of encoded video streams up to the picture corresponding to the OUT_time1 of Clip1 of Fig.90. Further, that video stream is connected to the video stream of the previous Clip1 and is re-encoded so that there results one continuous elementary stream in conformity with the MPEG2 standard.

[0275]

In a manner similar to the above, the video stream of Bridge-Clip subsequent to RSPN_arrival_time_discontinuity consists of an encoded video stream subsequent to a picture corresponding to IN_time2 of Clip2 of Fig.90. Further, that video stream is re-encoded so that decoding can be correctly made, and it is connected to the video stream of the Clip2 subsequent thereto so that there results one continuous elementary stream in conformity with the MPEG2 standard. For creating Bridge-Clip, several pictures must be, in general re-encoded, while pictures except for them can be copied from the original clip.

[0276]

An example of realizing seamless connection without using BridgeSequence in the case of the example shown in Fig.90 is shown in Fig.

92. The video stream of Clip1 consists of an encoded video stream up to a picture corresponding to OUT_time1 of Fig.90, this and that video stream is re-encoded so that there results one continuous elementary stream in conformity with the MPEG2 standard. Similarly, the video stream of Clip2 consists of an encoded video stream subsequent to the picture corresponding to the IN-time2 of Clip2 of Fig.90, and that stream is re-encoded so that there results one continuous elementary stream in conformity with the MPEG2 standard.

[0277]

The encoding limitation of the video stream will be explained. First, the frame rates of video streams of TS1 and TS2 must be equal to each other. The TS1 video stream must be terminated at sequence_end_code. The video stream of TS2 must be started at sequence header, GOP Header and I-picture. The video stream of TS2 must begins at closed GOP.

[0278]

The video presentation unit (frame or field) defined in a bitstream must be continuous with a connection point being put therebetween. It is prohibited that gap of frame or field exists at the connection point. At the connection point, field sequence of the top/bottom must be continuous. In the case of encoding employing the 3-2 pulldown, it may be necessary to rewrite "top_field_first" and "repeat_first_field" flag, or local re-encoding

may also be made so as to prevent occurrence of field gap.

[0279]

The encoding limitations of an audio bitstream will be explained. Audio sampling frequencies of TS1 and that of TS2 must be the same. The audio encoding methods of TS1 and TS2 ,e.g., MPEG1 layer 2, AC-3, SESF, LPCM and AAC must be the same.

[0280]

The encoding limitation of the MPEG-2 transport stream will now be explained. The last audio frame of the TS1 audio stream must includes an audio sample having a display time equal to display end time of the last display picture of TS1. The first audio frame of TS2 audio stream must includes an audio sample having a display time equal to that at display start time of the first display picture of TS2.

[0281]

At the connection point, there must not exist any gap in the sequence of the audio presentation unit. As shown in Fig. 93, there may be an overlap defined by the length of the audio presentation unit which is less than two audio frame period. The first packet for performing transmission of the elementary stream of TS2 must be video packet. The transport stream at the connection point must be in conformity with the DVR-STD which will be described later.

[0282]

The limitations of Clip and Bridge-Clip will be explained. TS1 and TS2 must not include discontinuous points of the arrival time bases in respective TS1 and TS2.

[0283]

The following limitations are applied only in the case where Bridge-Clip is used. The Bridge_Clip AV stream has discontinuous point of only one arrival time base only at the connection point between the last source packet of TS1 and the first source packet of TS2. The RSPN_arrival_time_discontinuity defined in ClipInfo() must indicate an address of its discontinuous point, and it must indicate the address which refers the first source packet of TS2.

[0284]

The source packet referred by the RSPN_exit_from_previous_Clip defined in BridgeSequenceInfo() may be any source packet in the Clip. It is unnecessary that this source packet is the boundary of the Aligned unit. The source packet referred by the RSPN_enter_to_current_Clip defined in BridgeSequenceInfo() may be any source packet in the Clip 2. It is unnecessary that such source packet is the boundary of the Aligned unit.

[0285]

The limitations on the PlayItem will be explained. The OUT_time of

previous PlayItem (OUT_time 1 shown in Figs.88 and 89) must indicate the display end time of the last video presentation unit of TS1. The IN_time of the current PlayItem (IN_time2 shown in Figs.88 and 89) must indicate the display start time of the first video presentation unit of TS2.

[0286]

The limitations on data allocation in the case where the Bridge-Clip is used will be explained with reference to Fig.102. The seamless connection must be created such as that continuous data supply guaranteed by the file system. This must be performed by disposing or assigning the Bridge-Clip AV stream, connected to the Clip1 (Clip AV stream file) and Clip2 (Clip AV stream file) so as to satisfy the data allocation prescriptions.

[0287]

The RSPN_exit_from_previous_Clip must be selected so that the stream portion of the Clip1 (Clip AV stream file) previous to RSPN_exit_from_previous_Clip (Clip AV stream file) is disposed or assigned within a continuous area havinghalf fragment or more. The data length of the Bridge-Clip AV stream must be selected so that the stream is disposed or assigned within a continuous area having half fragment or more. RSPN_enter_to_current_Clip must be selected so that the stream portion of Clip2 (Clip AV stream file) subsequent to RSPN_enter_to_current_Clip is disposed or assigned within a continuous area havinghalf fragment or more.

[0288]

The limitations on data allocation in the case where seamless connection is made without using Bridge-Clip will be explained. The seamless connection must be created so that continuous data supply is guaranteed by the file system. This must be carried out by disposing or assigning the first portion of last portion of the Clip1 (Clip AV stream file) and the first portion of the Clip2 (Clip AV stream file) so as to satisfy the data allocation prescriptions.

[0289]

The last stream portion of Clip1(Clip AV stream file) must be disposed or assigned within a continuous are having half fragment or more. The first stream portion of Clip2 (AV stream file) must be disposed or assigned within a continuous area having half fragment or more.

[0290]

The DVR-STD will now be explained. The DVR-STD is a conceptual model for modeling the decode processing in generation and verification of the DVR MPEG-2 transport stream. Moreover, the DVR-STD is also a conceptual model for modeling the decode processing in generation and verification of the AV stream refereed by the above-described seamless-connected two PlayItems.

[0291]

The DVR-STD model is shown in Fig. 96. In the model shown in Fig.96, there is included, as a component, a DVR MPEG-2 transport stream player model. The notations of n, Tbn, Mbn, Ebn, Tbsys, Bsys, Rxn, Rbxn, Rxsys, Dn, Dsys, On and Pn(k) are the same as those defined in T-STD of the ISO/IEC138188-1. Namely, the following description is provided. n is index number of elementary stream and Tbn is transport buffer of an elementary stream n.

[0292]

Mbn is a multiplex buffer of the elementary stream n, and exists only with respect to video stream. EBn is an elementary stream buffer in the elementary stream n. It exists only with respect to the video stream. TBsys is an input buffer for system information of program being decoded. Bsys is a main buffer within a system target decoder for the system information for a program being decoded. Rxn is transmission rate at which data is removed from TBn. Rbxn is transmission rate at which data is removed from MBn. It exists only with respect to video stream.

[0293]

Rxsys is transmission rate with which data is removed from TBsys. Dn is a decoder of elementary stream n. Dsys is s decoder relating to the system information of a program being decoded. On is a re-ordering buffer of the video stream n. Pn(k) is the k-th presentation unit of the elementary

stream n.

[0294]

The decoding process of the DVR-STD will be explained. For a time during which a single DVR MPEG-2 transport stream is reproduced, the timing at which the transport packet is inputted to the buffer of TBn or TBsys is determined by the arrival_time_stamp of the source packet. The prescriptions of buffering operations of TB1, MB1, EB1, TBn, Bn, TBsys and Bsys are the same as that of T-STD prescribed in the ISO/IEC B818-1. The prescriptions for the decoding operation and the display operation are the same as that of T-STD prescribed in ISO/IEC 13818-1.

[0295]

The decoding process for a time period during which the seamless-connected PlayItem is reproduced will be explained. Here, reproduction or playback of two AV streams, which is referred by the seamless_connected PlayItem, will be explained. In the following explanation, the reproduction or playback of the above-described TS1 And TS2, (e.g., shown in Fig.88) will be explained. TS1 and TS2 are a previous stream and a current stream, respectively.

[0296]

Fig.97 shows a timing chart of inputting, decode and display of transport packet when shift from a certain AV stream (TS1) to the subsequent

AV stream (TS2) seamlessly connected thereto is performed. For a time period during which shift from a predetermined AV stream (TS1) to the subsequent AV stream (TS2) connected seamlessly thereto is performed, the time axis of the arrival time base of TS2 (indicated at STC2 in Fig.97) is not the same as the time axis of the arrival time base of TS1 (indicated at STC1 in Fig.97).

[0297]

Moreover, the time axis of the system time base of TS2 (indicated at STC2 in Fig.97) is not the same as the time axis of the system time base of TS1 (indicated at STC1 in Fig.97). Video display is required to be seamlessly continued. There there may be overlap in the display time of the audio presentation unit.

[0298]

The input timing to the DVR-STD will be explained. Until the time T1, i.e., the time at which the last video packet of TS1 is completely inputted to the TB1 of DVR-STD, the input timing to the buffer of TB1, TBn or TBsys of DVR-STD is determined by the arrival_time_stamp of the source packet of TS1.

[0299]

The remaining packet of TS1 must be inputted to the buffer of TBn or TBsys at a bitrate of TS_recording_rate (TS1), where TS_recording_rate

(TS1) is a value of the TS_recording_rate defined in the ClipInfo() corresponding to the Clip1. The time at which the last byte of TS1 is inputted to the buffer is the time T_2. Accordingly, in the time period from time T_1 up to time T_2, arrival_time_stamp of the source packet is disregarded.

[0300]

If N1 is the number of bytes of the transport packet of TS1 subsequent to the last video packet of TS1, the time DT1 from time T_1 up to time T_2 is a time required until input of N1 bytes is completed at a bitrate of TS_recording_rate (TS1), and is calculated in accordance with the following equation:

$$DT1 = T_2 - T_1 = N1/TS\_recording\_rate\ (TS1).$$

For a time period from the time T1 up to the time T2, values of RXn and RXsys are both changed into the value of TS_recording_rate (TS1). The buffering operation except for this rule is the same as that of the T-STD.

[0301]

At time T_2, the arrival time clock counter is reset to the value of arrival_time_stamp of the first source packet of TS2. The input timing to the input timing of the TB1, TBn or TBsys is determined by the arrival_time_stamp of the source packet of TS2. The Rxn and Rxsys are both changed into values defined in T-STD.

[0302]

The additional audio buffering and the system data buffering will be explained. The audio decoder and the system decoder are required to have an additional buffer quantity (data quantity corresponding to about one second) in addition to the buffer volume defined in T-STD.

[0303]

The video presentation timing of video will be explained. The display of the video presentation unit must be continuous, via the connection point, without gap. It is assumed that STC1 is the time axis of the TS1 system time base (shown at STC1 in Fig.97), and STC2 is the time axis of the TS2 system time base (shown at STC2 in Fig.97). More precisely, STC2 begins from the time at which the first PCR of TS2 is inputted to T-STD.

[0304]

The offset between STC1 and STC2 is determined as follows. When it is assumed that PTS^1_{end} is PTS on STC1 corresponding to the last video presentation unit of TS1, PTS^2_{start} is PTS on STC2 corresponding to the first video presentation unit of TS2, and T_{pp} is the display period of the last video presentation unit of TS1, the offset STC_delta between two system time bases can be calculated in accordance with the following equation:

$$STC\_delta = PTS^1_{end} + T_{pp} - PTS^2_{start}$$

[0305]

The timing of the audio presentation will be explained. There may

exist any overlap of display timing of audio presentation unit at the connection point, and it is zero to 2 audio frames or less (see "audio overlap" shown in Fig.97). Which audio sample should be selected and re-synchronization of the display of the audio presentation unit to corrected time base after the connection point are set by the player side.

[0306]

The system time clocks of DVR-STD will be explained. The last audio presentation unit of TS1 is displayed at time T_5. The system time clocks may overlap with each other between time T_2 and time T_5. For this time period, in the DVR-STD, system time clock is switched between the old time base value (STC1) and the new time base value (STC2). The value of STC2 can be calculated in accordance with the following equation:

$$STC2 = STC1 - STC\_delta.$$

[0307]

The buffering continuity will be explained. $STC^1_{video\_end}$ is the value of STC on the system time base STC1 when the last byte of the last video packet of TS1 reaches TB1 of DVR-STD. $STC^2_{video\text{-}start}$ is a value of STC on system time base when the first byte of the first video packet of TS2 reaches TB1 of DVR-SFD. $STC2^1_{video\_end}$ is a value converted into a value on the system time base STC2. The $STC2^1_{video\_end}$ is calculated by the following equation:

$$STC2^{1}_{video\_end} = STC1^{1}_{video\_end} - STC\_delta.$$

[0308]

In order to become in conformity with the DVR-STD, it is required to satisfy the following two conditions. First, the arrival timing to TB1 of first video packet of TS2 must satisfy the following inequality:

$$STC2^{2}_{video\_start} > STC2^{1}_{video\_end} + \Delta T_1.$$

In the case where there is a necessity to re-encode or re-multiplex partial stream of Clip 1 and/or Clip 2 in order that the above-mentioned inequality is satisfied, such operation is performed as occasion demands.

[0309]

Then, on the time axis of the system time base in which STC1 and STC2 are converted onto the same time axis, input of video packet from TS1 and input of video packet from TS2 subsequent thereto is prohibited to allow the video buffer to overflow or underflow.

[0310]

On the basis of such syntax, data structure and rule, it is possible to suitably perform management of the content of data recorded on the recording medium and/or reproduction information . Thus, user can suitably confirm the content of data recorded on the recording medium at the time of reproduction or playback, on or to reproduce desired data with ease.

[0311]

While the above-described sequence of operations can be executed by hardware, they may be also executed by software. In the case where a sequence of processing is caused to be performed by the software, programs constituting the software are installed from a recording pedium into a computer in which those programs are assembled in a dedicated hardware, and/or, e.g., general-purpose personal computer in which various programs are installed to thereby have ability to execute various functions, etc.

[0312]

As shown in Fig.98, this recording medium not only may be constituted by a package medium comprised of a magnetic disc 221 (including of floppy disc), an optical disc 222 (including CD-ROM (Compact Disc–Read Only memory) and DVD (Digital versatile Disc)), a magneto-optical disc 223 (including MD (Mini-Disc)) and or a semiconductor memory 224, etc. where programs are recorded, which is distributed for the purpose of offering program to user, but also may be constituted by a hard disc, etc. in which a ROM 202 or a memory 208 having program stored therein, which is offered to user in the state assembled into the computer in advance.

[0313]

In this specification, steps which describe the program offered by a medium includes not only the processing executed in time series manner in accordance with the described order, but also includes processing executed in

parallel or individually even if they are not necessarily processed in time series manner.

[0314]

In addition, in this specification, the system represents the entirety of an apparatus composed of plural devices.

[0315]

[Advantages/Effects of the invention]

As described above, in accordance with the information processing apparatus according to claim 1, the information processing method according to claim 3 and the recording medium according to claim 4, since AV stream which has been recorded from the time point where recording of AV stream has been started up to the time point where it is completed is caused to be one unit to record, onto or into recording medium, at least one of information relating to recording mode of AV stream, information relating to average value of recording rate of the AV stream, information relating to the period where encoded information is the same of the AV stream, information relating to random accessible position in the AV stream and informating relating to featured image in the AV stream as attribute information of the AV stream, on the unit basis, such information may be used, thereby making it possible to rapidly perform determination of readout position and/or decode processing of AV stream.

[Brief Description of the Drawings]

Fig. 1 is a view showing the configuration of an embodiment of a recording/reproducing apparatus to which the present invention is applied.

Fig. 2 is a view for explaining the format of data recorded on a recording medium by the recording/reproducing apparatus 1.

Fig. 3 is a view for explaining Real PlayList and Virtual PlayList.

Fig. 4 is a view for explainign the creation of the Real PlayList.

Fig. 5 is a view for explaining deletion of the Real PlayList.

Fig. 6 is a view for explaining assemble editing.

Fig. 7 is a view for explaining the case where sub path is provided in the Virtual PlayList.

Fig. 8 is a view for explaining change of the playback order of the PlayList.

Fig. 9 is a view for explaining mark on the PlayList and mark on the Clip.

Fig. 10 is a view for explaining menu thumbnail.

Fig. 11 is a view for explaining mark added to the PlayList.

Fig. 12 is a view for explaining mark added to the Clip.

Fig. 13 is a view for explaining the relationship between the PlayList, Clip and the thumbnail file.

Fig. 14 is a view for explaining directory structure.

Fig. 15 is a view showing syntax of infr.dvr.

Fig. 16 is a view showing syntax of DVR volume.

Fig. 17 is a view showing syntax of ResumeVolume.

Fig. 18 is a view showing syntax of UIAppInfoVolume.

Fig. 19 is a view showing table of Character set value.

Fig. 20 is a view showing syntax of TableOfPlayList.

Fig. 21 is a view showing another syntax of TableOfPlayList.

Fig. 22 is a view showing syntax of the MakersPricvateData.

Fig. 23 is a view showing syntax of xxxx.rpls and yyyyy.vpls.

Fig. 24 is a view for explaining the PlayList.

Fig. 25 is a view showing syntax of PlayList.

Fig. 26 is a view showing table of PlayList_type.

Fig. 27 is a view showing syntax of UIAppInfoPlayList.

Fig. 28 is a view for explaining flags in the syntax of the UIAppInfoPlayList shown in Fig. 27.

Fig. 29 is a view for explaining PlayItem.

Fig. 30 is a view for explaining PlayItem.

Fig. 31 is a view for explaining PlayItem.

Fig. 32 is a view showing syntax of the PlayItem.

Fig. 33 is a view for explaining IN_time.

Fig. 34 is a view for explaining OUT_time.

Fig. 35 is a view showing table of Connection _Condition.

Fig. 36 is a view for explaining Connection_Condition.

Fig. 37 is a view for explaining BridgeSequenceInfo.

Fig. 38 is a view showing syntax of BridgeSequenceInfo.

Fig. 39 is a view for explaining SubPlayItem.

Fig. 40 is a view showing syntax of SubPlayItem.

Fig. 41 is a view showing table of SubPath_type.

Fig. 42 is a view showing syntax of PlayListMark.

Fig. 43 is a view showing table of Mark_type.

Fig. 44 is a view for explaining Mark_time_stamp.

Fig. 45 is a view showing syntax of zzzzz.clip.

Fig. 46 is a view showing syntax of ClipInfo.

Fig. 47 is a view showing table of Clip_stream_type.

Fig. 48 is a view for explaining offset_SPN.

Fig. 49 is a view for explaining offset_SPN.

Fig. 50 is a view for explaining the STC period.

Fig. 51 is a view for explaining STC_Info.

Fig. 52 is a view showing syntax of STC_Info.

Fig. 53 is a view for explaining ProgramInfo.

Fig. 54 is a view showing syntax of ProgramInfo.

Fig. 55 is a view showing syntax of VideoCondingInfo.

Fig. 56 is a view showing table of Video_format.

Fig. 57 is a view showing table of frame_rate.

Fig. 58 is a view showing table of display_aspect_ratio.

Fig. 59 is a view showing syntax of AudioCondingInfo.

Fig. 60 is a view showing table of audio_coding.

Fig. 61 is a view showing table of audio_component_type.

Fig. 62 is a view showing table of sampling_frequency.

Fig. 63 is a view for explaining CPI.

Fig. 64 is a view for explaining CPI.

Fig. 65 is a view showing syntax of CPI.

Fig. 66 is a view showing table of CPI_type.

Fig. 67 is a view for explaining video EP_map.

Fig. 68 is a view for explaining EP_map.

Fig. 69 is a view for explaining EP_map.

Fig. 70 is a view showing syntax of EP_map.

Fig. 71 is a view showing table of EP_type values.

Fig. 72 is a view showing syntax of EP_map_for_one_stream_PID.

Fig. 73 is a view for explaining TU_map.

Fig. 74 is a view showing syntax of TU_map.

Fig. 75 is a view showing syntax of ClipMark.

Fig. 76 is a view showing table of mark_type.

Fig. 77 is a view showing table of mark_type_stamp.

Fig. 78 is a view showing syntax of menu.thmb and mark.thmb.

Fig. 79 is a view showing syntax of Thumbnail.

Fig. 80 is a view showing table of thumbnail_picture_format.

Fig. 81 is a view for explaining tn_block.

Fig. 82 is a view for explaining the structure of a transport stream of DVR MPEG2.

Fig. 83 is a view showing recorder model of the transport stream of DVR MPEG2.

Fig. 84 is a view showing player model of the transport stream of DVR MPEG2.

Fig. 85 is a view showing syntax of source packet.

Fig. 86 is a view showing syntax of TP_extra_header.

Fig. 87 is a view showing table of copy permission indicator.

Fig. 88 is a view for explaining seamless connection.

Fig. 89 is a view for explaining seamless connection.

Fig. 90 is a view for explaining seamless connection.

Fig. 91 is a view for explaining seamless connection.

Fig. 92 is a view for explaining seamless connection.

Fig. 93 is a view for explaining overlap of audio.

Fig. 94 is a view for explaining seamless connection using

BridgeSequence.

Fig. 95 is a view for explaining seamless connection without using BridgeSequence.

Fig. 96 is a view showing DVR STD model.

Fig. 97 is a view showing a timing chart for decoding and display.

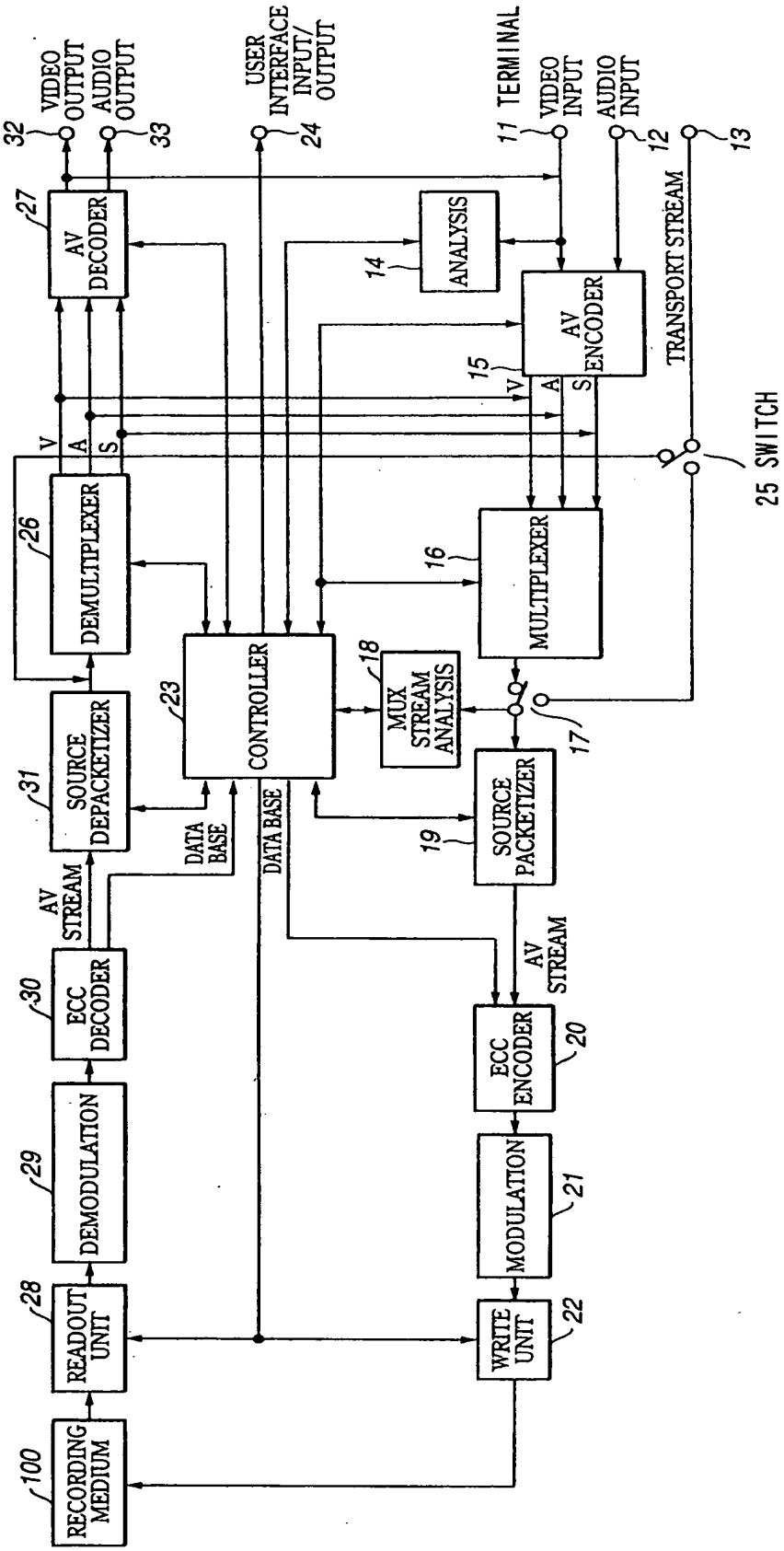
Fig. 98 is a view for explaining medium.

[Explanation of referenced numbers]

1 Recording/Reproducing Apparatus, 11 to 13 Terminal, 14 Analysis Unit, 15 AV Encoder, 16 Multiplexer, 17 Switch, 18 Multiplexed Stream Analysis Unit, 19 Source Packetizer, 20 ECC Unit, 21 Modulation Unit, 22 Write Unit, 23 Controller, 24 User Interface, 25 Switch, 26 Demultiplexer, 27 AV Decoder, 28 Readout Unit, 29 Demodulation Unit, 30 ECC Decoding Unit, 31 Source Packetizer, 32, 33 Terminal

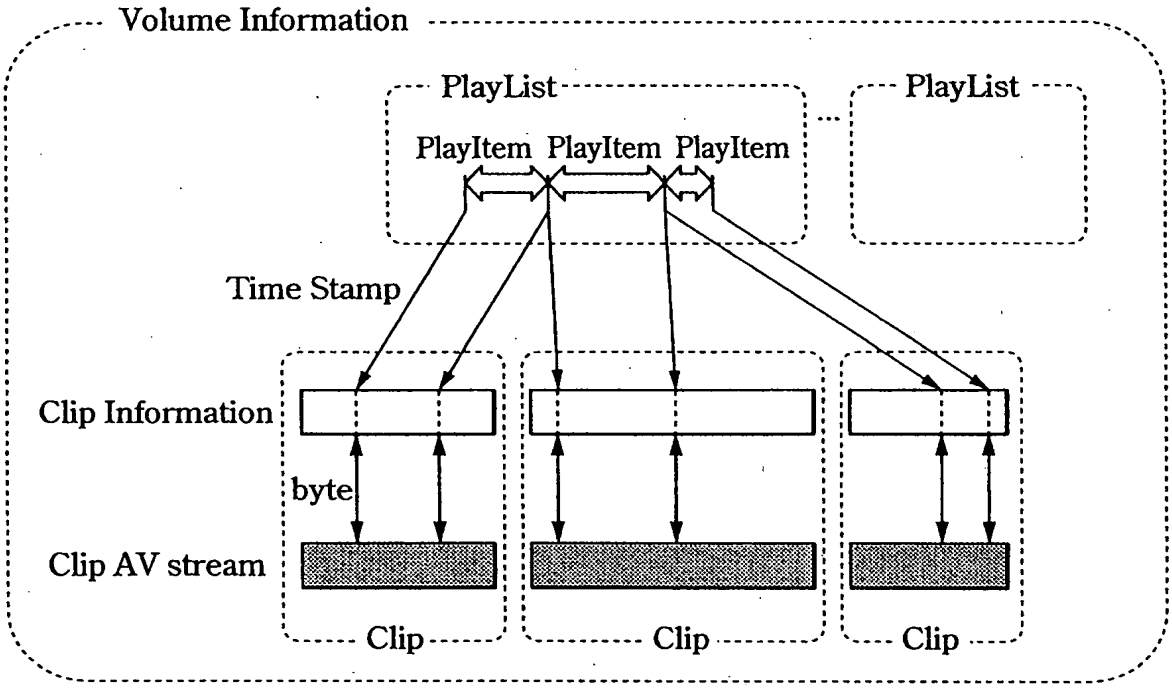
[DOCUMENT NAME] DRAWING

[FIG.1]

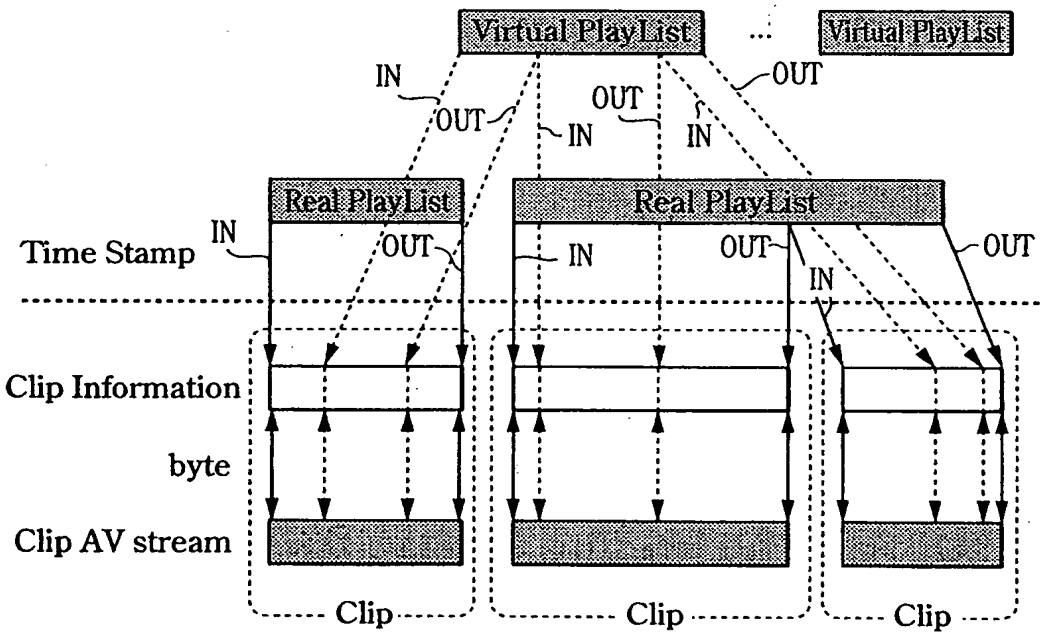


Recording and/or reproducing apparatus 1

[FIG.2]

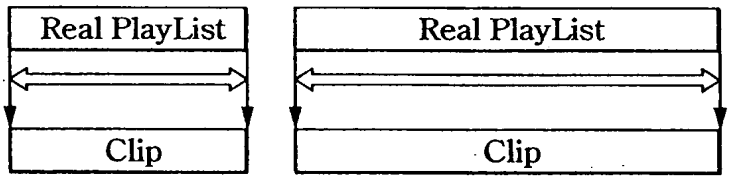

Volume Information
PlayList
PlayList
PlayItem
PlayItem
PlayItem
Time Stamp
Clip Information
byte
Clip AV stream
Clip
Clip
Clip

[FIG.3]


Virtual PlayList
...
Virtual PlayList
IN
OUT
OUT
IN
OUT
IN
Real PlayList
Real PlayList
Time Stamp
IN
OUT
IN
OUT
IN
OUT
Clip Information
byte
Clip AV stream
Clip
Clip
Clip

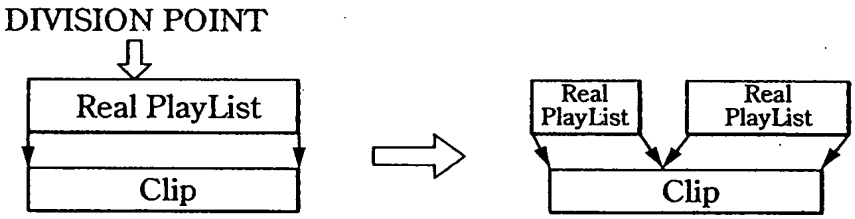
[FIG.4]

(A)



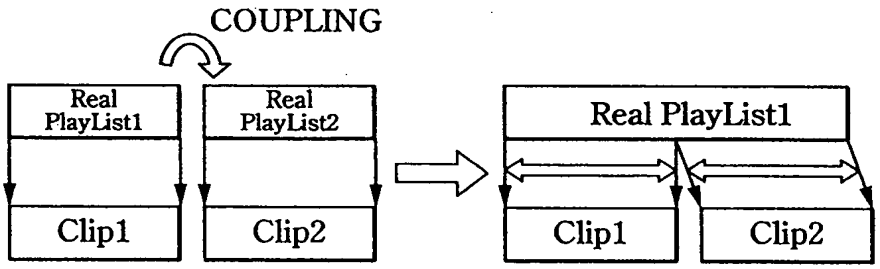
Creation of Real PlayList

(B)



Dividing Real PlayList

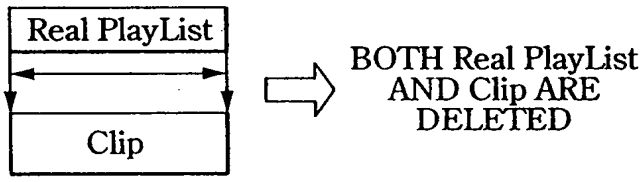
(C)



Combining Real PlayList

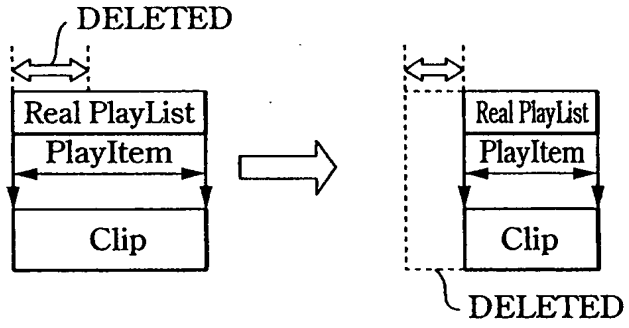
[FIG.5]

(A)



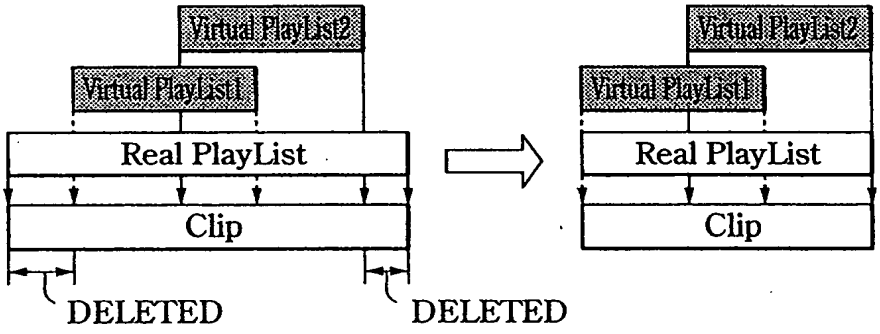
Deletion of entire Real PlayList

(B)



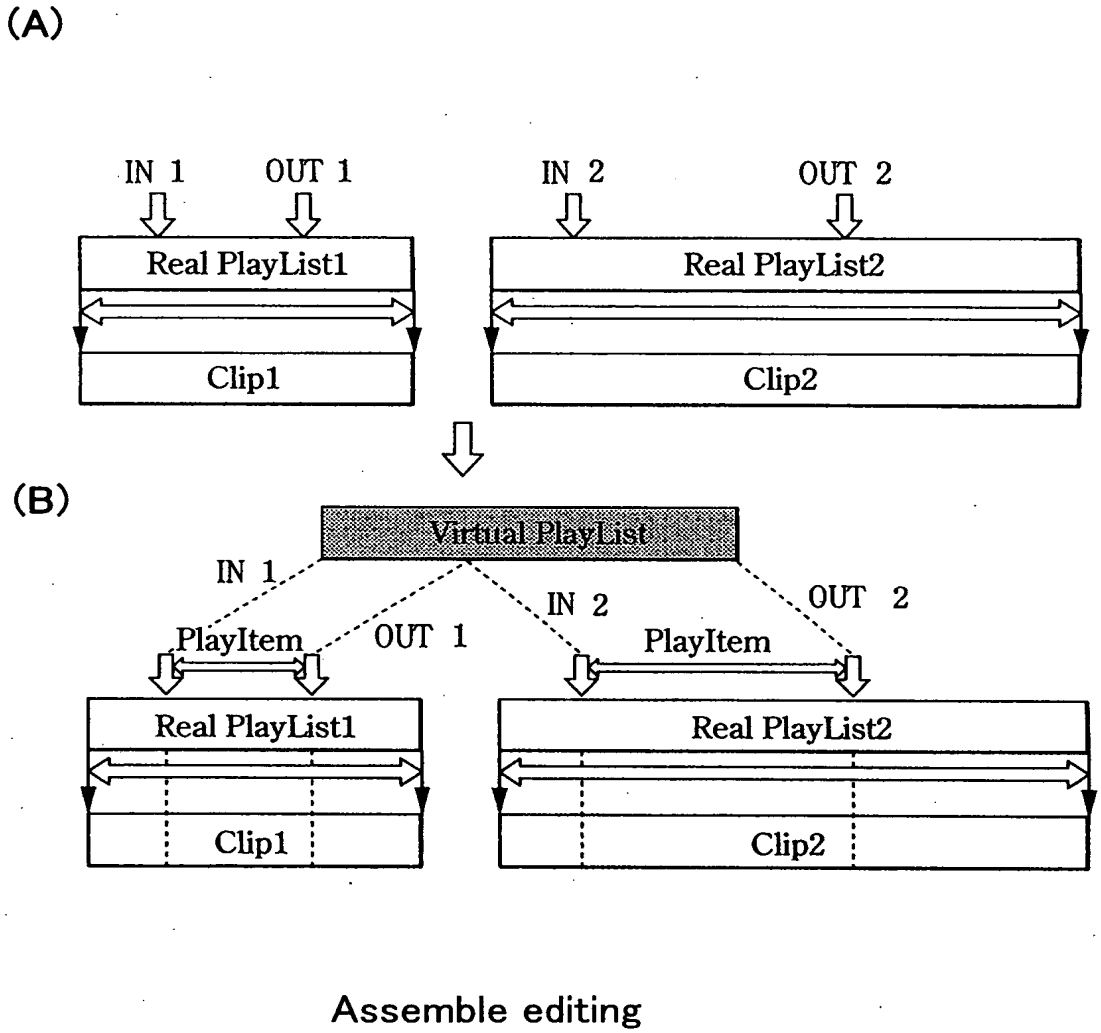
Partial deletion of Real PlayList

(C)



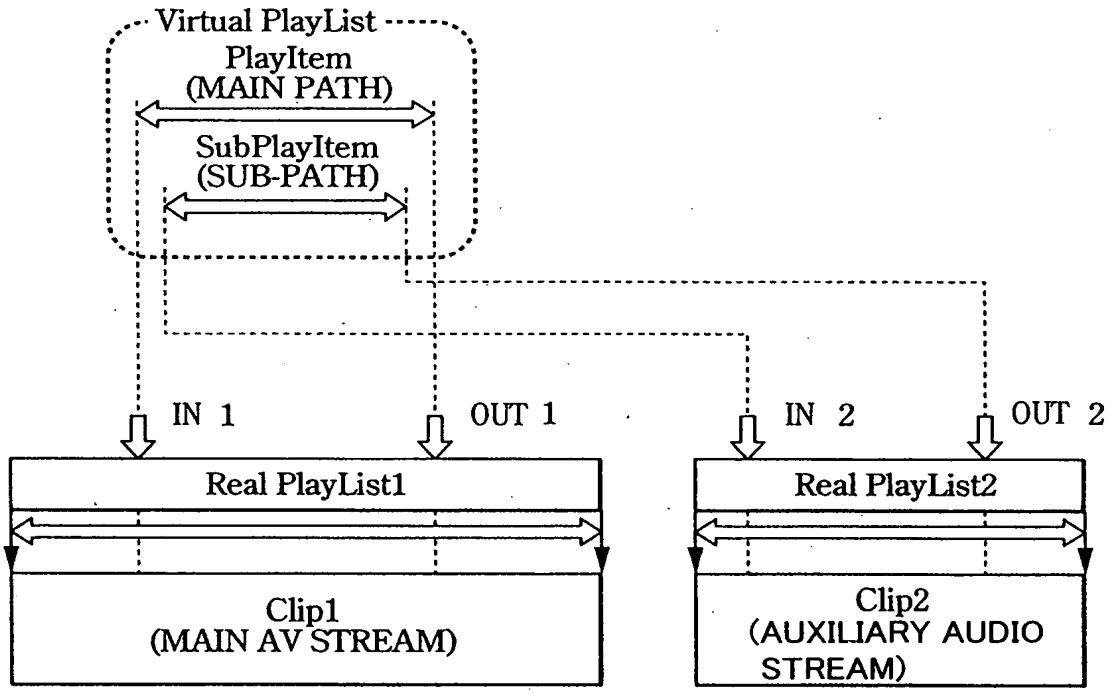
Minimizing of Real PlayList

[FIG.6]



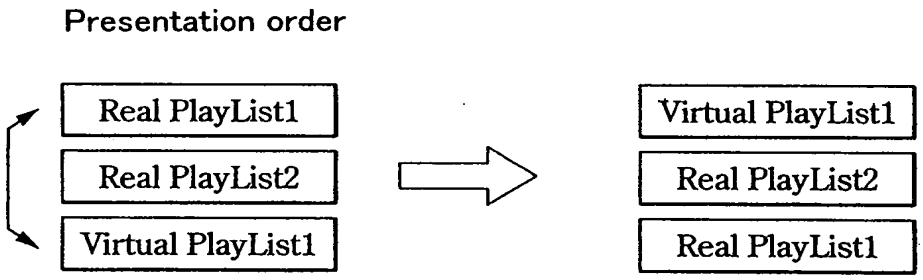
Assemble editing

[FIG.7]



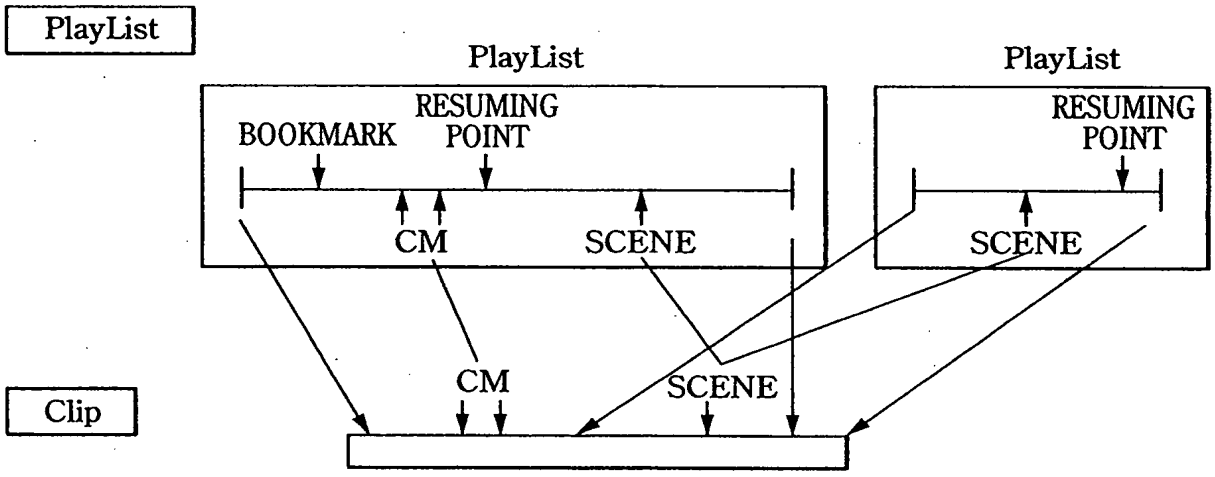
Audio dubbing to Virtual PlayList

[FIG.8]



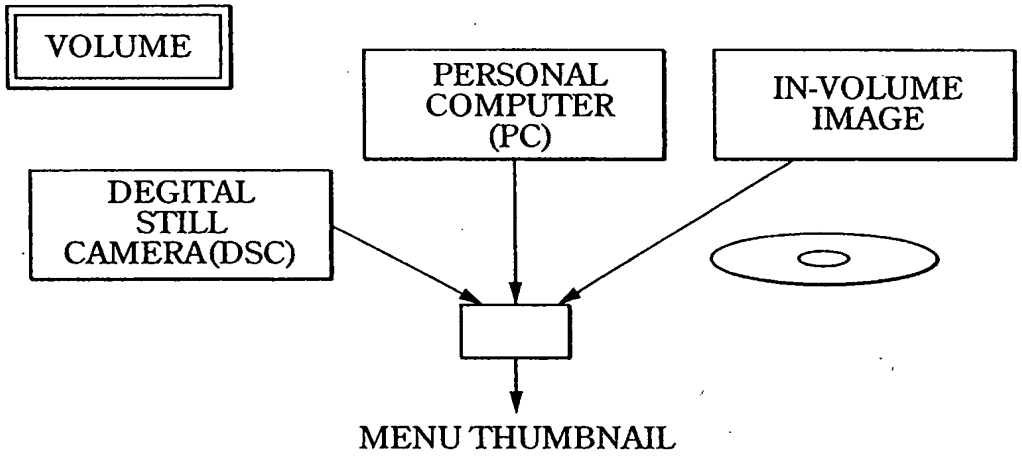
Changing of playback sequence of PlayList

[FIG.9]

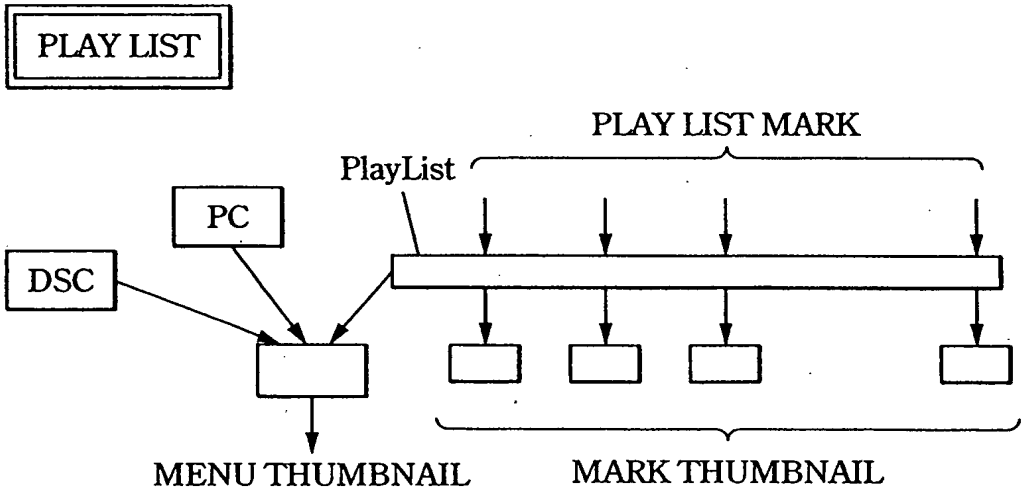


Mark on PlayList and Mark on Clip

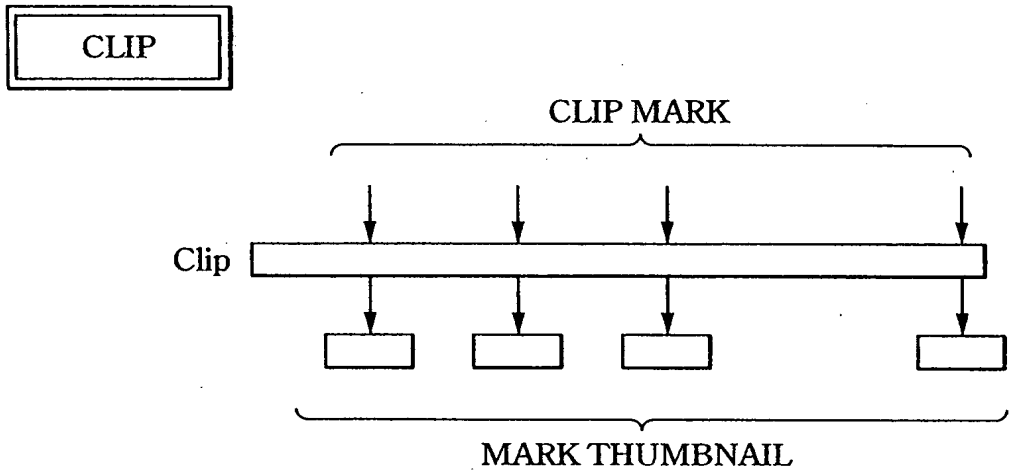
[FIG.10]


VOLUME
PERSONAL COMPUTER (PC)
IN-VOLUME IMAGE
DEGITAL STILL CAMERA(DSC)
MENU THUMBNAIL

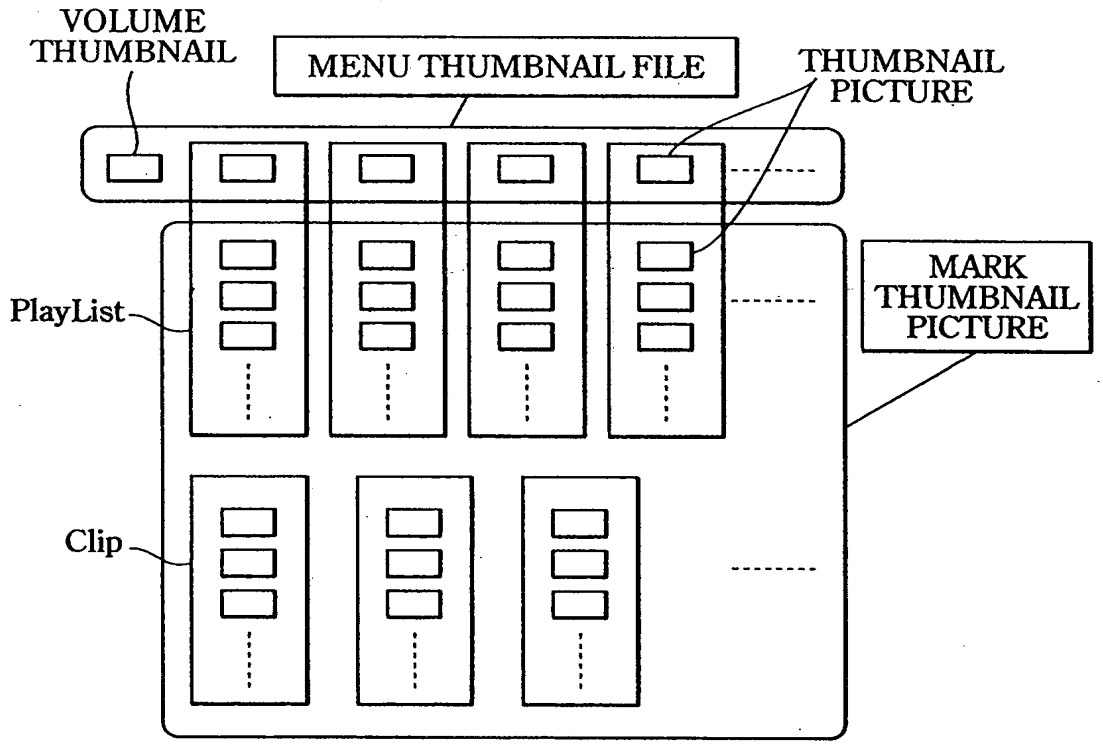
[FIG.11]


PLAY LIST
PLAY LIST MARK
PlayList
PC
DSC
MENU THUMBNAIL
MARK THUMBNAIL

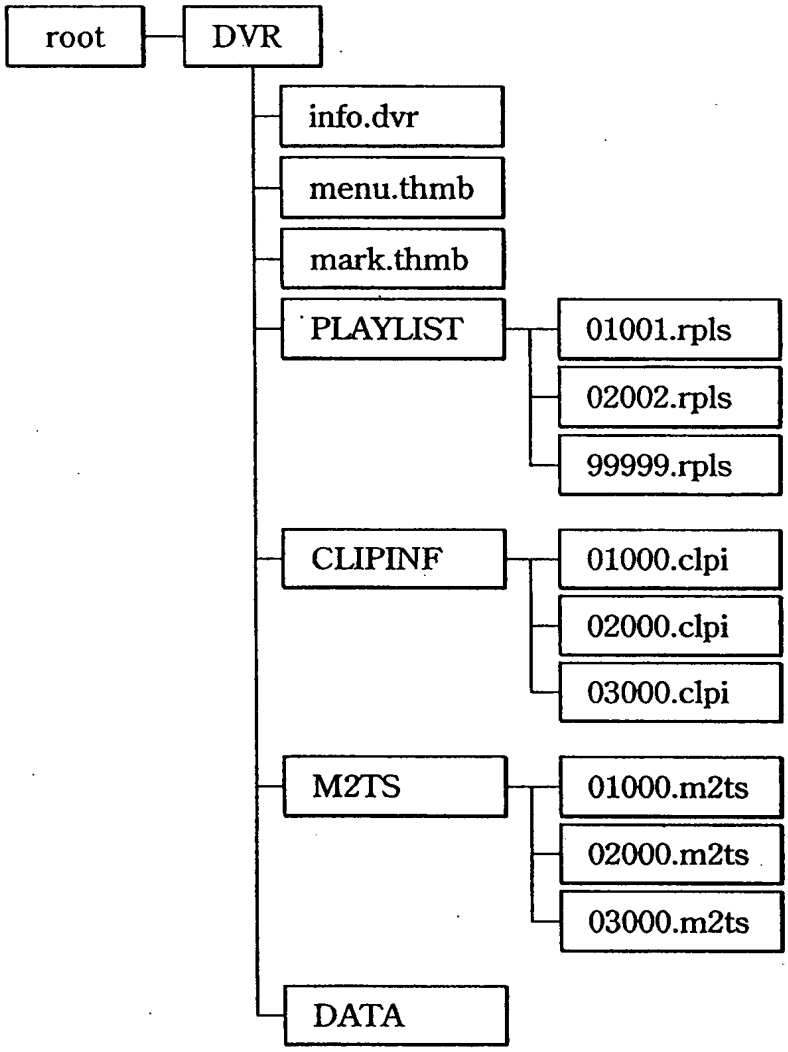
[FIG.12]


CLIP
CLIP MARK
Clip
MARK THUMBNAIL

[FIG.13]


VOLUME THUMBNAIL
MENU THUMBNAIL FILE
THUMBNAIL PICTURE
MARK THUMBNAIL PICTURE
PlayList
Clip

[FIG.14]


root
DVR
info.dvr
menu.thmb
mark.thmb
PLAYLIST
01001.rpls
02002.rpls
99999.rpls
CLIPINF
01000.clpi
02000.clpi
03000.clpi
M2TS
01000.m2ts
02000.m2ts
03000.m2ts
DATA

[FIG.15]

SYNTAX	No. of bits	Mnemonics
info.dvr {		
TableOfPlayLists_Start_address	32	uimsbf
MakersPrivateData_Start_address	32	uimsbf
reserved	192	bslbf
DVRVolume()		
for (i=0;i<N1;i++){		
padding_word	16	bslbf
}		
TableOfPlayLists()		
for (i=0;i<N2;i++){		
padding_word	16	bslbf
}		
MakersPrivateData()		
}		

Syntax of infr.dvr

[FIG.16]

SYNTAX	No. of bits	Mnemonics
DVRVolume(){		
version_number	8*4	bslbf
length	32	uimsbf
ResumeVolume()		
UIAppInfoVolume()		
}		

Syntax of DVRVolume

[FIG.17]

SYNTAX	No. of bits	Mnemonics
ResumeVolume(){		
reserved	15	bslbf
valid_flag	1	bslbf
resume_PlayList_name	8*10	bslbf
}		

Syntax of ResumeVolume

[FIG.18]

SYNTAX	No. of bits	Mnemonics
UIAppInfoVolume(){		
character_set	8	bslbf
name_length	8	uimsbf
Volume_name	8*256	bslbf
reserved	15	bslbf
Volume_protect_flag	1	bslbf
PIN	8*4	bslbf
ref_thumbnail_index	16	uimsbf
reserved_for_future_use	256	bslbf
}		

Syntax of UIAppInfoVolume

[FIG.19]

VALUE	CHARACTER LETTER ENCODING
0x00	Reserved
0x01	ISO/IEC 646 (ASCII)
0x02	ISO/IEC 10646-1 (Unicode)
0x03-0xff	Reserved

Character set values

[FIG.20]

SYNTAX	No. of bits	Mnemonics
TableOfPlayLists(){		
version_number	8*4	bslbf
length	32	uimsbf
number_of_PlayLists	16	uimsbf
for (i=0; i<*number_of_PlayLists*; i++){		
PlayList_file_name	8*10	bslbf
}		
}		

Syntax of TableOfPlayList

[FIG.21]

TableOfPlayLists – syntax (another example of 4.2.3.2)

SYNTAX	No. of bits	Mnemonics
TableOfPlayLists(){		
version_number	8*4	bslbf
length	32	uimsbf
number_of_PlayLists	16	uimsbf
for (i=0; i<*number_of_PlayLists*; i++){		
PlayList_file_name	8*10	bslbf
UIAppInfoPlayList()		
}		
}		

Another syntax of TableOfPlayLists

[FIG.22]

SYNTAX	No. of bits	Mnemonics
MakersPrivateData(){		
version_number	8*4	bslbf
length	32	uimsbf
if (length !=0){		
mpd_blocks_start_address	32	uimsbf
number_of_maker_entries	16	uimsbf
mpd_block_size	16	uimsbf
number_of_mpd_blocks	16	uimsbf
reserved	16	bslbf
for (i=0; i<*number_of_maker_entries*; i++){		
maker_ID	16	uimsbf
maker_model_code	16	uimsbf
start_mpd_block_number	16	uimsbf
reserved	16	bslbf
mpd_length	32	uimsbf
}		
stuffing_bytes	8*2*L1	bslbf
for(j=0; j<*number_of_mpd_blocks*; j++){		
mpd_block	mpd_block_size*1024*8	
}		
}		
}		

Syntax of MakersPrivateData

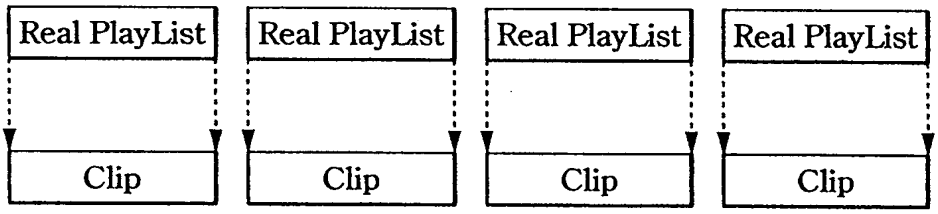
[FIG.23]

SYNTAX	No. of bits	Mnemonics
xxxxx.rpls / yyyyy.vpls {		
PlayListMark_Start_address	32	uimsbf
MakersPrivateData_Start_address	32	uimsbf
reserved	192	bslbf
PlayList()		
for (i=0;i<N1;i++){		
padding_word	16	bslbf
}		
PlayListMark()		
for (i=0;i<N2;i++){		
padding_word	16	bslbf
}		
MakersPrivateData()		
}		

Syntax of xxxx.rpls and yyyy.vpls

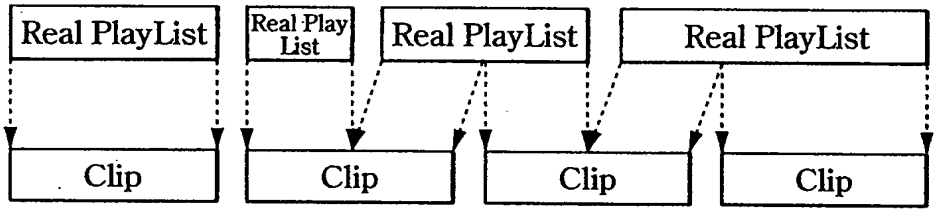
[FIG.24]

(A)



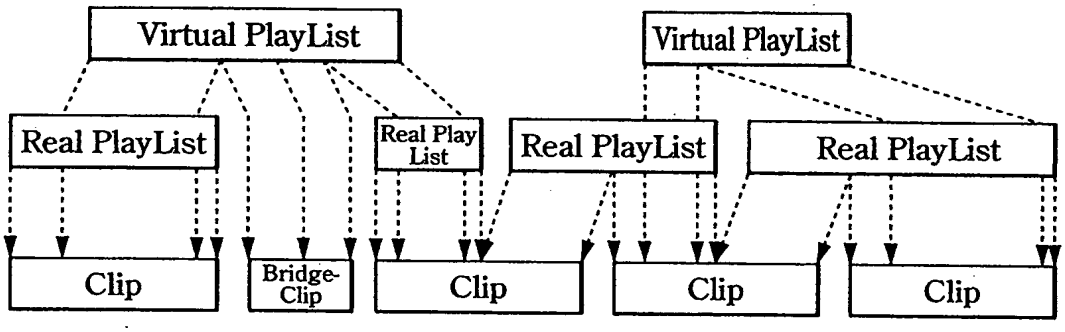
Real PlayList when AV stream is firstly recorded as Clip

(B)



Real PlayList after editing operation

(C)



Virtual PlayList

[FIG.25]

SYNTAX	No. of bits	Mnemonics
PlayList(){		
version_number	8*4	bslbf
length	32	uimsbf
PlayList_type	8	uimsbf
CPI_type	1	bslbf
reserved	7	bslbf
UIAppInfoPlayList()		
number_of_PlayItems // main path	16	uimsbf
if (<*Vertual PlayList*>){		
number_of_SubPlayItems // sub path	16	uimsbf
}else{		
reserved	16	bslbf
}		
for (*PlayItem_id=0;* *PlayItem_id<nymber_of_PlayItems;* *PlayItem_id++*){		
PlayItem() //main path		
}		
if (<*Virtual PlayList*>){		
if (CPI_type==0 && PlayList_type==0){		
for (i=0; i<*number_of _SubPlayItems*; i++)		
SubPlayItem() //sub path		
}		
}		
}		

Syntax of PlayList

[FIG.26]

PlayList_type	MEANING
0	PLAY LIST FOR AV RECORDING ALL CLIPS REFERENCED IN THIS PLAY LIST MUST CONTAIN ONE OR MORE VIDEO STREAMS
1	PLAY LIST FOR AUDIO RECORDING ALL CLIPS REFERENCED IN THIS PLAYLIST MUST CONTAIN ONE OR MORE AUDIO STREAMS AND MUST NOT CONTAIN VIDEO STREAMS
2-255	reserved

PlayList_type

[FIG.27]

SYNTAX	No. of bits	Mnemonics
UIAppInfoPlayList2(){		
character_set	8	bslbf
name_length	8	uimsbf
PlayList_name	8*256	bslbf
reserved	8	bslbf
record_time_and_date	4*14	bslbf
reserved	8	bslbf
duration	4*6	bslbf
valid_period	4*8	bslbf
maker_id	16	uimsbf
maker_code	16	uimsbf
reserved	11	bslbf
playback_control_flag	1	bslbf
write_protect_flag	1	bslbf
is_played_flag	1	bslbf
archive	2	bslbf
ref_thumbnail_index	16	uimsbf
reserved_for_future_use	256	bslbf
}		

Syntax of UIAppInfoPlayList

[FIG.28]

(A)

write_protect_flag	MEANING
0b	THE PlayList CAN BE ERASED FREELY
1b	THE PlayList CONTENTS SHOULD NOT BE ERASED NOR CHANGED EXCEPT write-protect-flag

write_protect_flag

(B)

is_played_flag	MEANING
0b	THE PlayList HAS NOT BEEN REPRODUCED SINCE ITS RECORDING
1b	THE PlayList WAS ONCE REPRODUCED SINCE ITS RECORDING

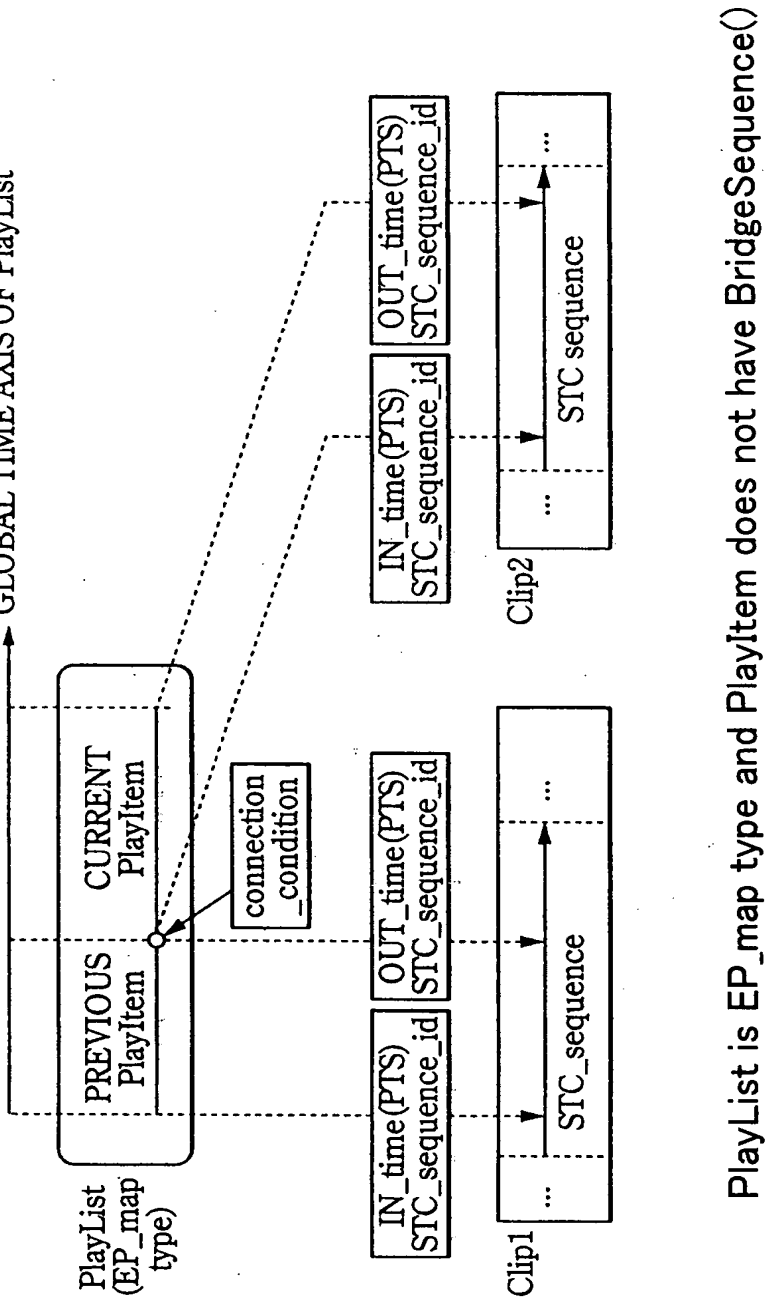
is_played_flag

(C)

archive	MEANING
00b	NO MEANING DEFINED
01b	ORIGINAL
10b	COPY
11b	reserved

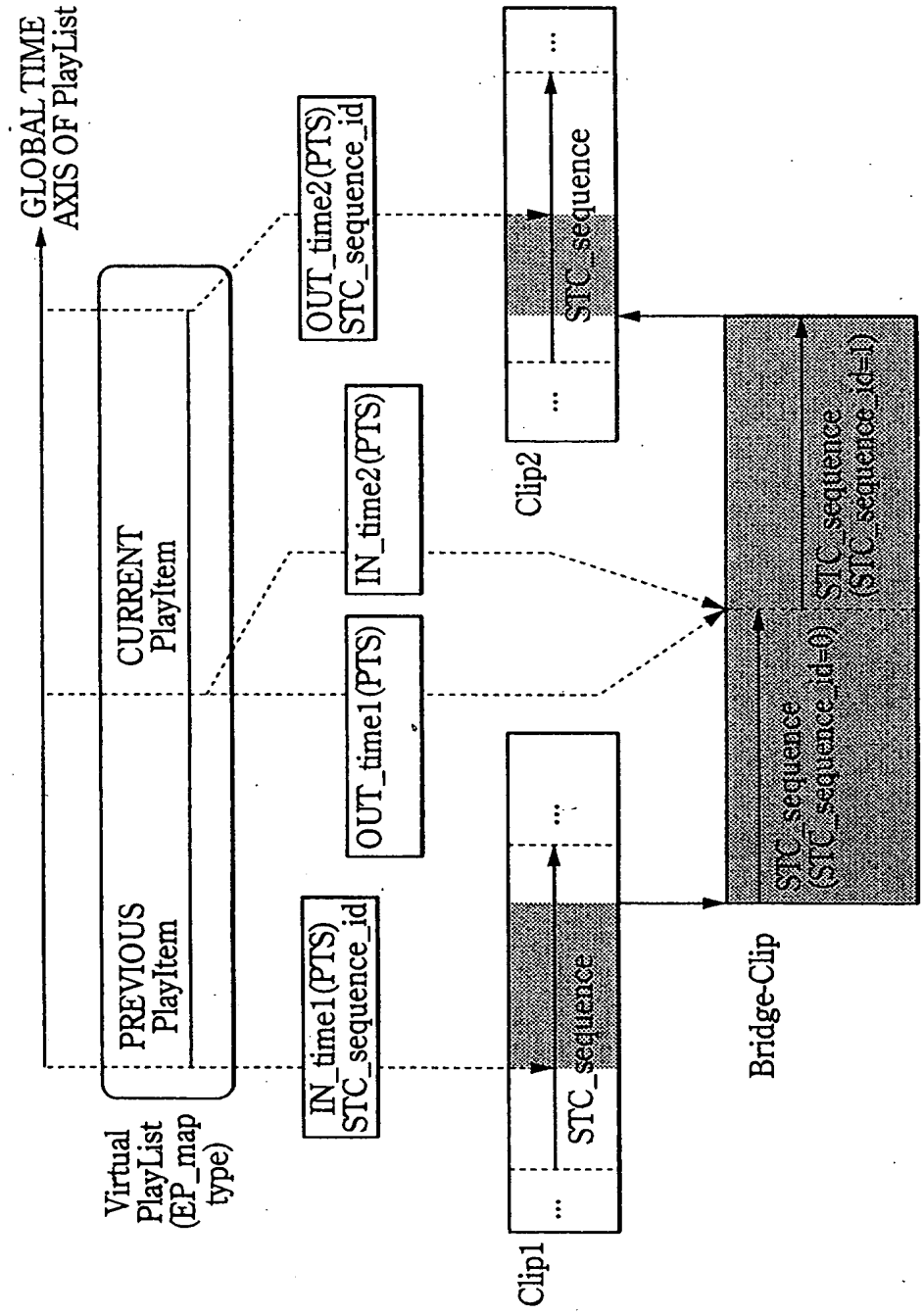
archive

[FIG.29]



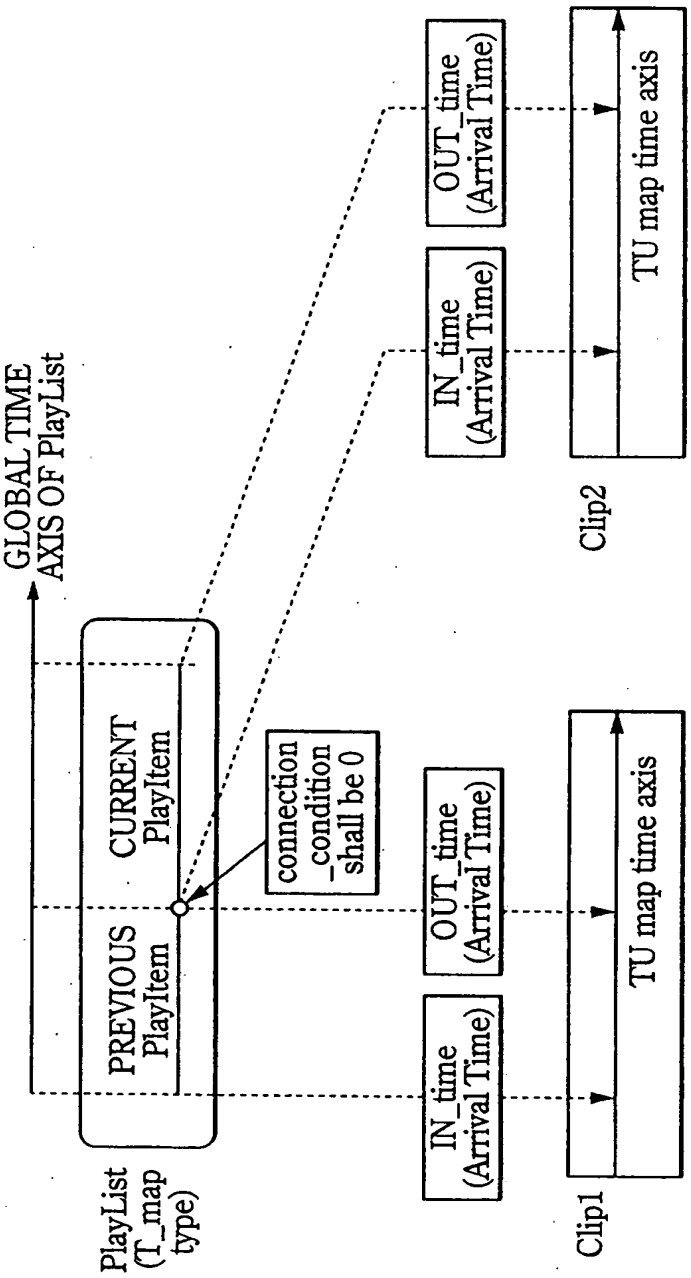
PlayList is EP_map type and PlayItem does not have BridgeSequence()

[FIG.30]



PlayList is EP_map type and PlayItem has BridgeSequence()

[FIG.31]


GLOBAL TIME AXIS OF PlayList
PlayList (T_map type)
PREVIOUS PlayItem
CURRENT PlayItem
connection _condition shall be 0
IN_time (Arrival Time)
OUT_time (Arrival Time)
Clip1
TU map time axis
IN_time (Arrival Time)
OUT_time (Arrival Time)
Clip2
TU map time axis
PlayList() is TU_map type

[FIG.32]

SYNTAX	No. of bits	Mnemonics
PlayItem(){		
Clip_information_file_name	8*10	bslbf
reserved	24	bslbf
STC_sequence_id	8	uimsbf
IN_time	32	uimsbf
OUT_time	32	uimsbf
reserved	14	bslbf
connection_condition	2	bslbf
if (<Virtual PlayList>){		
if (*connection_condition*=='*10*'){		
BridgeSequenceInfo()		
}		
}		
}		

Syntax of PlayItem

[FIG.33]

CPI_type in the PlayList()	SEMANTICS OF IN_time
EP_map type	IN_time MUST INDICATE UPPER 32 BITS OF 33 BIT LENGTH CORRESPONDING TO FIRST PRESENTATION UNIT IN PlayItem
TU_map type	IN_time MUST BE TIME ON TU_map_time_axis, AND MUST BE ROUNDED TO time_unit PRECISION. IN-time IS CALCULATED BY FOLLOWING EQUATION: $IN\_time = TU\_start\_time \%2^{32}$

IN-time

[FIG.34]

CPI_type in the PlayList()	SEMANTICS OF OUT_time
EP_map type	OUT_time MUST INDICATE UPPER 32 BITS OF THE VALUE OF Presentation_end_TS CALCULATED BY FOLLOWING EQUTION: Presentation_end_TS = PTS_out+AU_duration WHERE PTS_out IS 33-BIT LONG PTS CORRESPONDING TO LAST PRESENTATION UNIT IN PlayItem. AU_duration IS 90 kHz-DISPLAY TIME OF LAST PRESENTATION UNIT.
TU_map type	OUT_time MUST BE TIME ON TU_map_time_axis AND BE ROUNDED TO time_unit PRECISION. OUT_time IS CALCULATED BY FOLLOWING EQUATION: $OUT\_time = TU\_start\_time \ \%2^{32}$

OUT-time

[FIG.35]

connection _condition	MEANING
00	· CONNECTION OF PREVIOUS PlayItem TO CURRENT PlayItem IS NOT SURE AS TO SEAMLESS REPLAY. · IF CPI_type OF PlayList IS TU_map type, THIS VALUE MUST BE SET IN connection_condition.
01	· THIS STATE IS ALLOWED ONLY WHEN CPI_type OF PlayList IS EP_map type. · PREVIOUS PlayItem AND CURRENT PlayItem INDICATE DIVISION BECAUSE OF NON-CONTINUOUS POINT OF SYSTEM TIMEBASE (STC BASE).
10	· THIS STATE IS ALLOWED ONLY WHEN CPI_type OF PlayList IS EP_map type. · THIS STATE IS ALLOWED ONLY FOR Virtual PlayList. · CONNECTION OF PREVIOUS PlayItem TO CURRENT PlayItem IS SURE AS TO SEAMLESS REPLAY. · PREVIOUS PlayItem IS CONNECTED TO CURRENT PlayItem USING BridgeSequence. DVR MPEG-2 TRANSPORT STREAM MUST OBEY DVR-STD AS LATER DESCRIBED.
11	· THIS STATE IS ALLOWED ONLY WHEN CPI_type OF PlayList IS EP_map type. · CONNECTION OF PREVIOUS PlayItem TO CURRENT Play Item IS SURE AS TO SEAMLESS REPLAY. · PREVIOUS PlayItem IS CONNECTED TO CURRENT PlayItem WITHOUT USING BridgeSequence. DVR MPEG-2 TRANSPORT STREAM MUST OBEY DVR-STD AS LATER DESCRIBED.

connection_condition

[FIG.36]

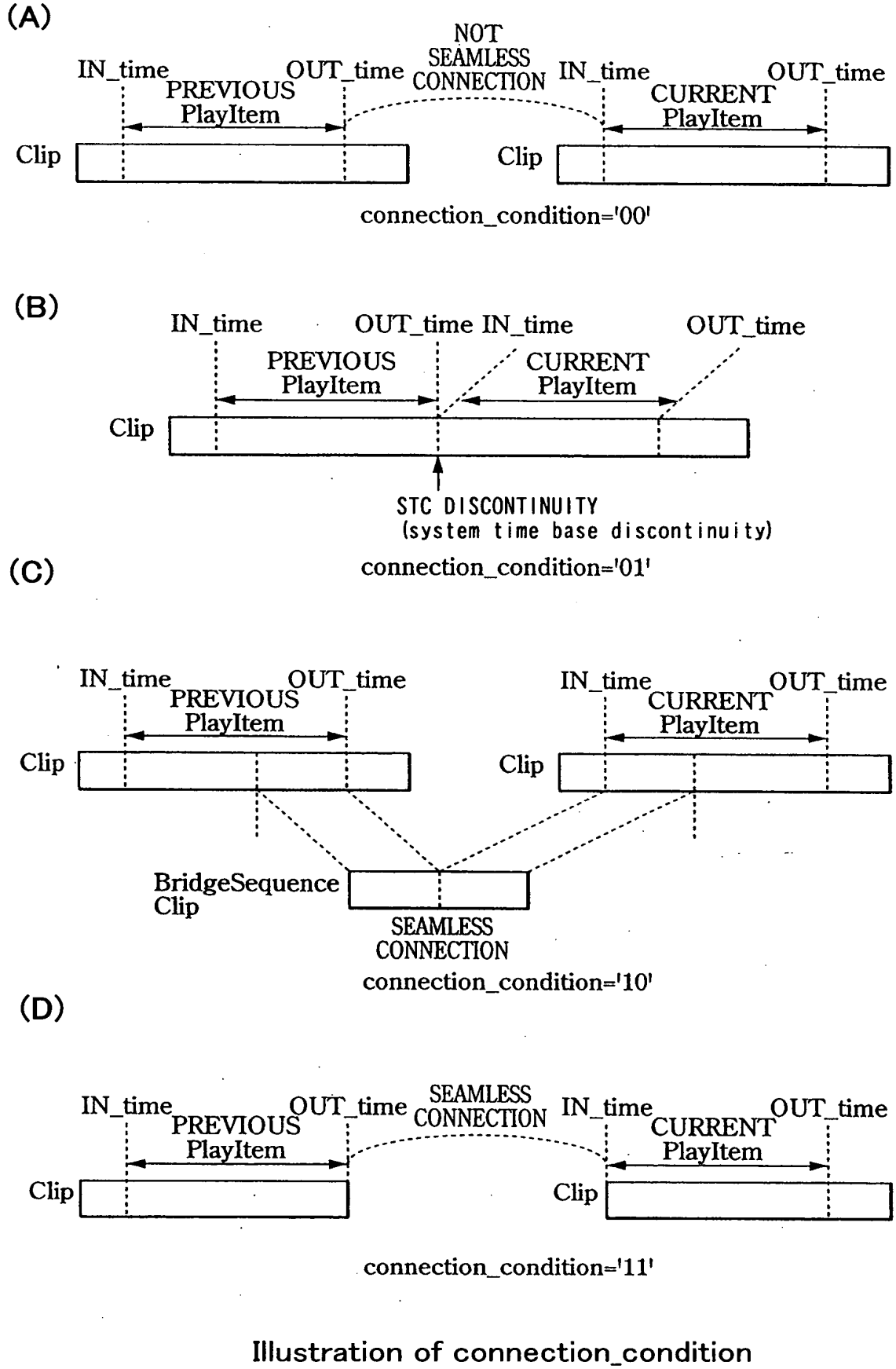
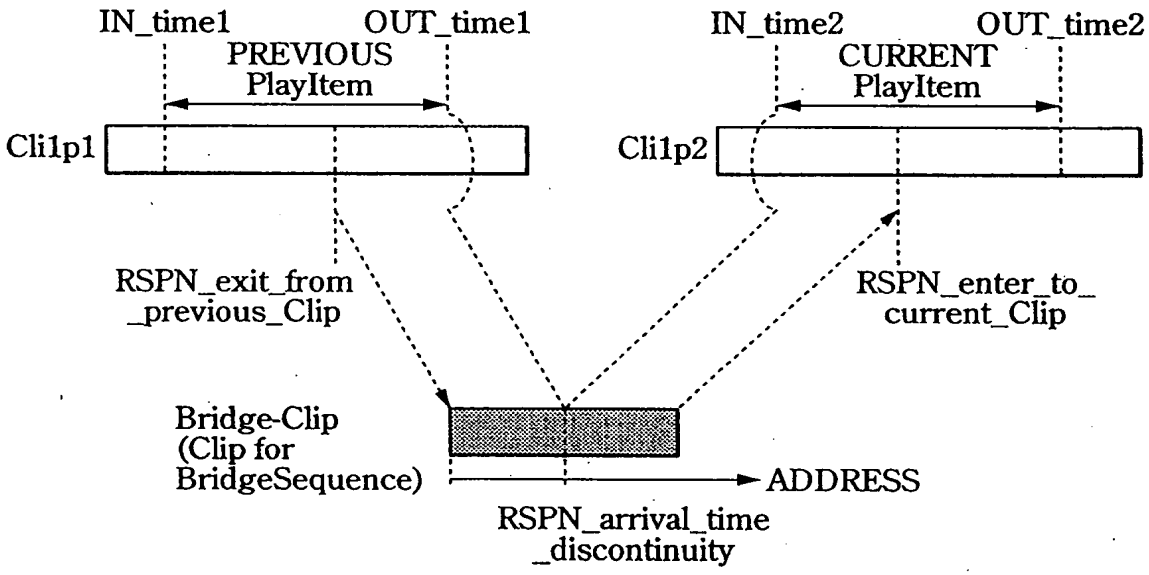


Illustration of connection_condition

[FIG.37]

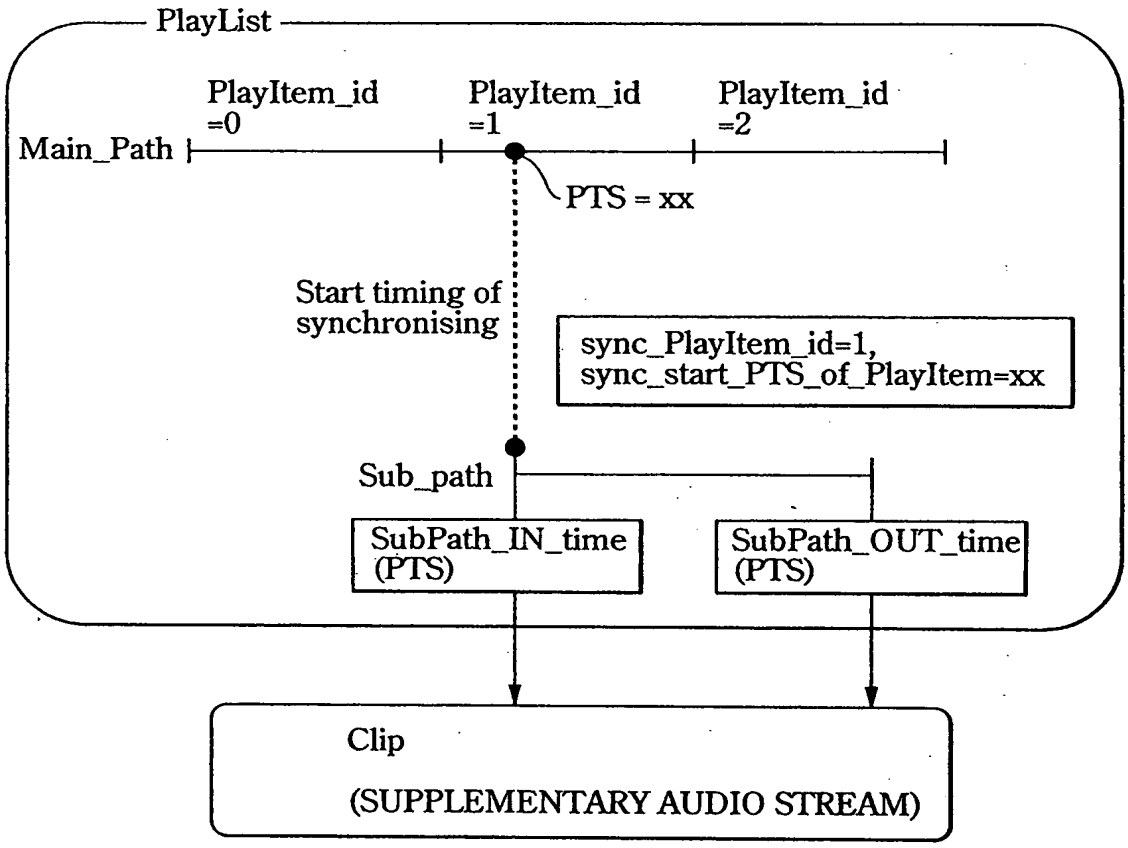

IN_time1
OUT_time1
PREVIOUS PlayItem
Clip1
IN_time2
OUT_time2
CURRENT PlayItem
Clip2
RSPN_exit_from _previous_Clip
RSPN_enter_to_ current_Clip
Bridge-Clip (Clip for BridgeSequence)
ADDRESS
RSPN_arrival_time _discontinuity

[FIG.38]

SYNTAX	No. of bits	Mnemonics
BridgeSequenceInfo() {		
Bridge_Clip_information_file_name	8*10	bslbf
RSPN_exit_from_previous_Clip	32	uimsbf
RSPN_enter_to_current_Clip	32	uimsbf
}		

Syntax of BridgeSequenceInfo

[FIG.39]


PlayList
PlayItem_id =0
PlayItem_id =1
PlayItem_id =2
Main_Path
PTS = xx
Start timing of synchronising
sync_PlayItem_id=1, sync_start_PTS_of_PlayItem=xx
Sub_path
SubPath_IN_time (PTS)
SubPath_OUT_time (PTS)
Clip
(SUPPLEMENTARY AUDIO STREAM)

[FIG.40]

SYNTAX	No. of bits	Mnemonics
SubPlayItem(){		
Clip_Information_file_name	8*10	bslbf
SubPath_type	8	bslbf
sync_PlayItem_id	8	uimsbf
sync_start_PTS_of_PlayItem	32	uimsbf
SubPath_IN_time	32	uimsbf
SubPath_OUT_time	32	uimsbf
}		

Syntax of SubPlayItem

[FIG.41]

SubPath_type	MEANING
0x00	Auxiliary audio steam path
0x01-0xff	reserved

SubPath_type

[FIG.42]

SYNTAX	No. of bits	Mnemonics
PlayListMark(){		
version_number	8*4	bslbf
length	32	uimsbf
number_of_PlayList_marks	16	uimsbf
for (i=0;i<*number_of_PlayList_marks*;i++){		
reserved	8	bslbf
mark_type	8	bslbf
mark_time_stamp	32	uimsbf
PlayItem_id	8	uimsbf
reserved	24	uimsbf
character_set	8	bslbf
name_length	8	uimsbf
mark_name	8*256	bslbf
ref_thumbnail_index	16	uimsbf
}		
}		

Syntax of PlayListMark

[FIG.43]

Mark_type	MEANING	COMMENT
0x00	resume-mark	REPLAY RESUME POINT. THE NUMBER OF REPLAY RESURE POINTS DEFINED IN PlayListMark() MUST BE 0 OR 1.
0x01	book-mark	REPLAY ENTRY POINT OF PlayList. THIS MARK CAN BE SET BY USER AND USED AS MARK SPECIFYING START POINT OF FAVORITE SCENE.
0x02	skip-mark	SKIP MARK POINT. PLAYER SKIPS PROGRAM FROM THIS POINT TO THE END OF PROGRAM. THE NUMBER OF SKIP MARK POINTS DEFINED IN PlayListMark() MUST BE 0 RO 1.
0x03-0x8F	reserved	
0x90-0xFF	reserved	Reserved for ClipMark()

mark_type

[FIG.44]

CPI_type in the PlayList()	SEMANTICS OF mark_time_stamp
EP_map type	mark_time_stamp MUST INDICATE UPPER 32 BITS OF 33 BIT LENGTH PTS CORRESPONDING TO PRESENTATION UNIT REFERENCED BY MARK.
TU_map type	mark_time_stamp MUST BE TIME ON TU_map_time_axis AND MUST BE ROUNDED TO time_unit PRECISION. mark_time_stamp IS CALCULATED BY FOLLOWING EQUATION: $mark\_time\_stamp = TU\_start\_time \ \%2^{32}$

mark_time_stamp

[FIG.45]

SYNTAX	No. of bits	Mnemonics
zzzzz.clpi {		
STC_Info_Start_address	32	uimsbf
ProgramInfo_Start_address	32	uimsbf
CPI_Start_address	32	uimsbf
ClipMark_Start_address	32	uimsbf
MakersPrivateData_Start_address	32	uimsbf
reserved	96	bslbf
ClipInfo()		
for (i=0;i<N1;i++){		
padding_word	16	bslbf
}		
STC_Info()		
for (i=0;i<N2;i++){		
padding_word	16	bslbf
}		
ProgramInfo()		
for (i=0;i<N3;i++){		
padding_word	16	bslbf
}		
CPI()		
for (i=0;i<N4;i++){		
padding_word	16	bslbf
}		
ClipMark()		
for (i=0;i<N5;i++){		
padding_word	16	bslbf
}		
MakersPrivateData()		
}		

Syntax of zzzzz.cipi

[FIG.46]

SYNTAX	No. of bits	Mnemonics
ClipInfo(){		
version_number	8*4	bslbf
length	32	uimsbf
Clip_stream_type	8	bslbf
offset_SPN	32	uimsbf
TS_recording_rate	24	uimsbf
reserved	8	bslbf
record_time_and_date	4*14	bslbf
reserved	8	bslbf
duration	4*6	bslbf
reserved	7	bslbf
time_controlled_flag	1	bslbf
TS_average_rate	24	uimsbf
if (Clip_stream_type==1) // Bridge-Clip AV stream		
RSPN_arrival_time_discontinuity	32	uimsbf
else		
reserved	32	bslbf
reserved_for_system_use	144	bslbf
reserved	11	bslbf
is_format_identifier_valid	1	bslbf
is_original_network_ID_valid	1	bslbf
is_transport_stream_ID_valid	1	bslbf
is_service_ID_valid	1	bslbf
is_country_code_valid	1	bslbf
format_identifier	32	bslbf
original_network_ID	16	uimsbf
transport_stream_ID	16	uimsbf
service_ID	16	uimsbf
country_code	24	bslbf
stream_format_name	16*8	bslbf
reserved_for_fortune_use	256	bslbf
}		

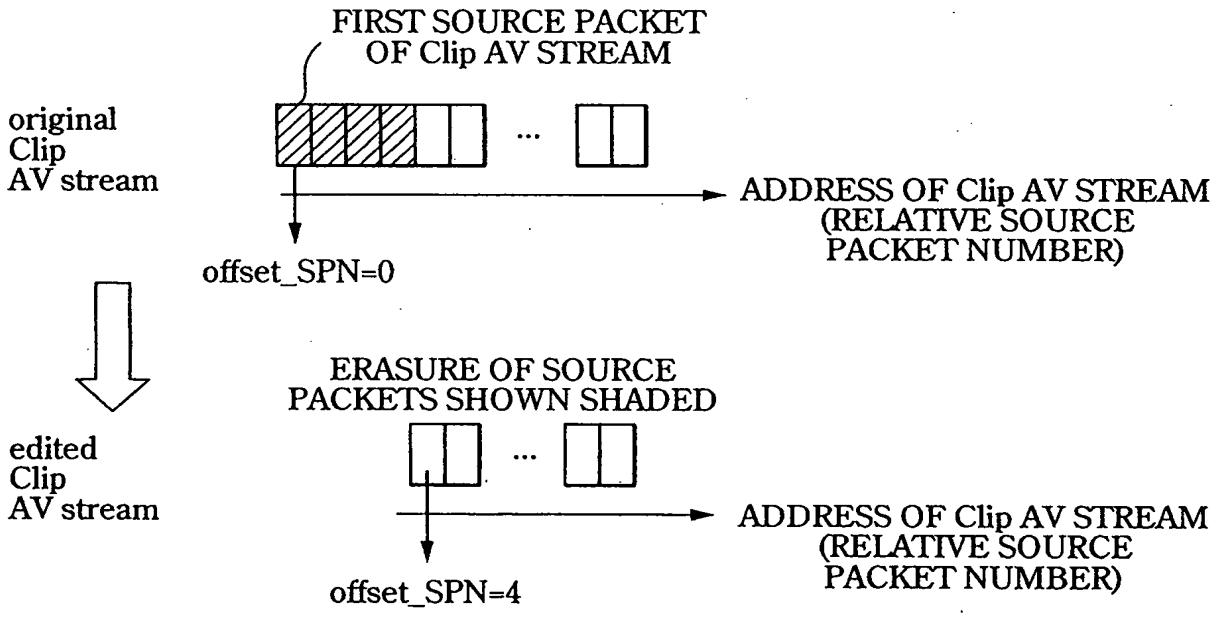
Syntax of ClipInfo

[FIG.47]

Clip_stream_type	MEANING
0	Clip AV STREAM
1	Bridge-Clip AV STREAM
2-255	Reserved

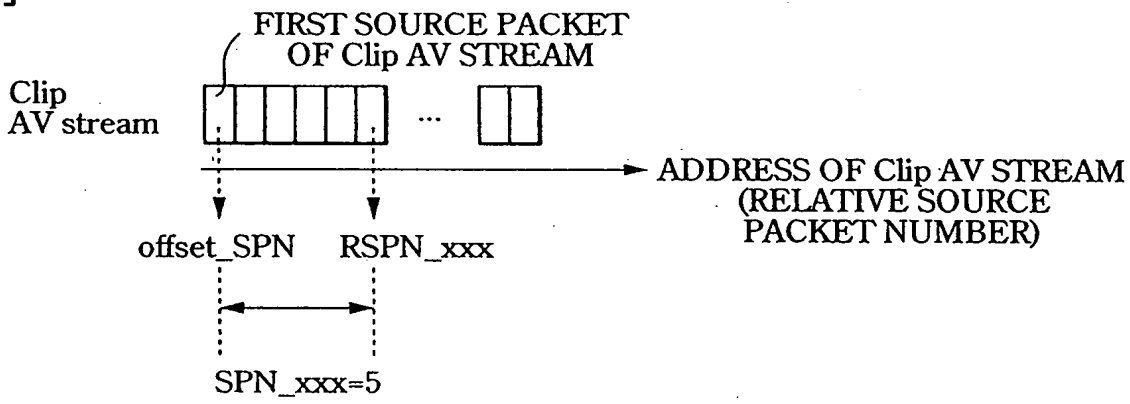
Clip_stream_type

[FIG.48]



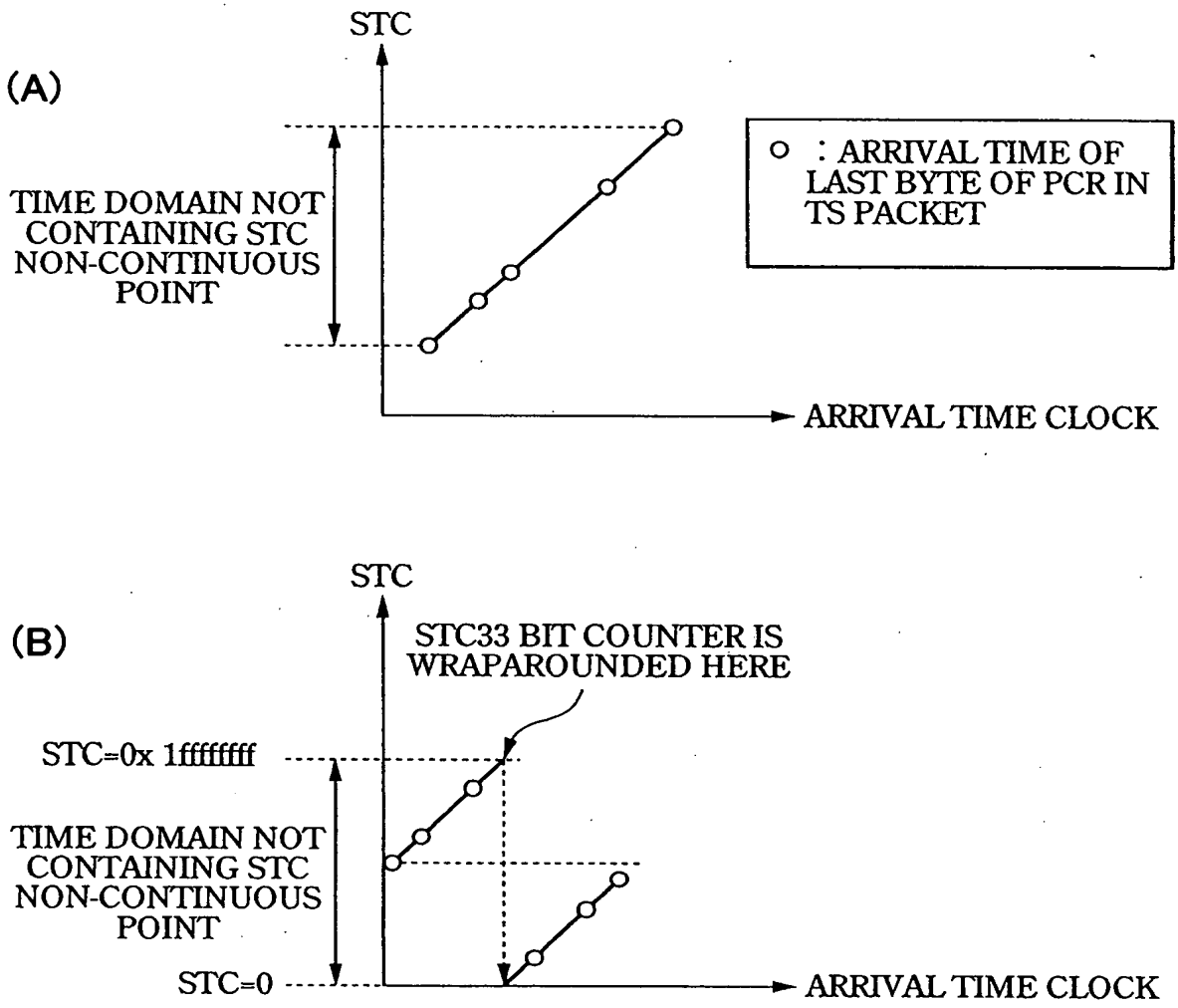
offset_SPN assumes value other than 0

[FIG.49]

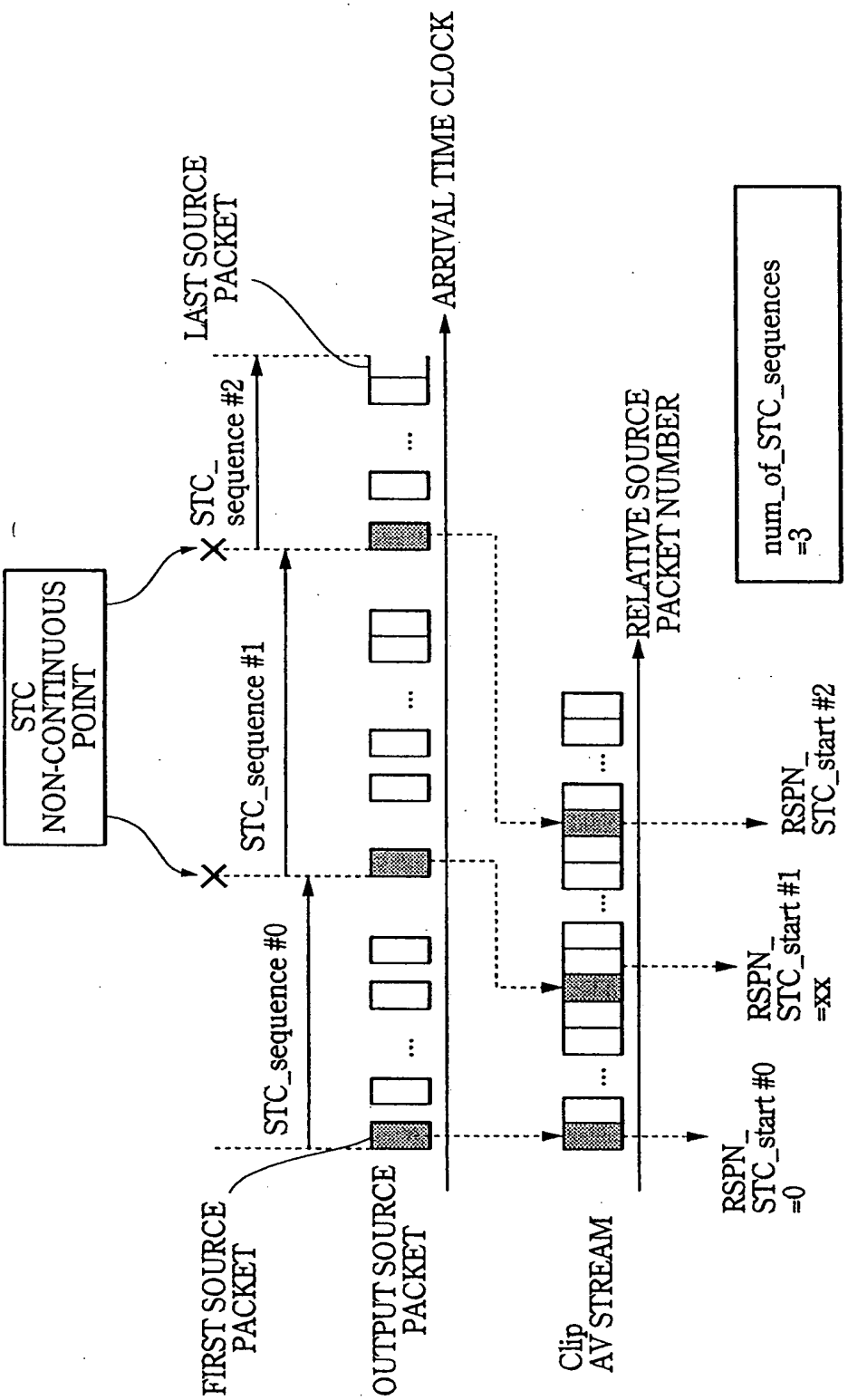


Relation between offset_SPN and relative source packet number (SPN_xxx) in AV stream

[FIG.50]



[FIG.51]



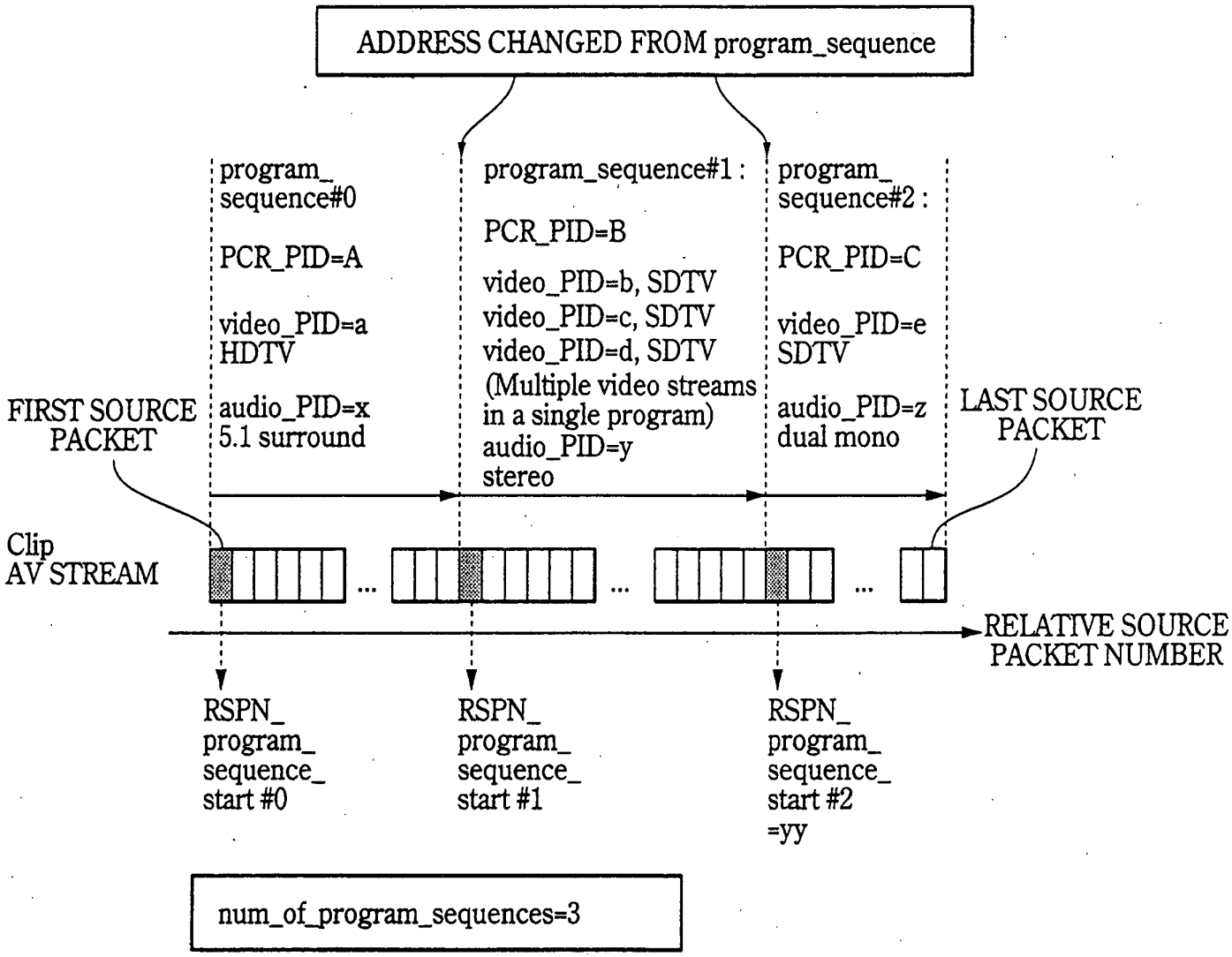
STC_Info

[FIG.52]

SYNTAX	No. of bits	Mnemonics
STC_Info(){		
version_number	8*4	bslbf
length	32	uimsbf
if (length !=0){		
reserved	8	bslbf
num_of_STC_sequences	8	uimsbf
for (*STC_sequence_id=0;* *STC_sequence_id<num_of_STC_sequences;* *STC_sequence_id++*){		
resereved	32	bslbf
RSPN_STC_start	32	uimsbf
}		
}		
}		

Syntax of STC_Info

[FIG.53]



ProgramInfo

[FIG.54]

SYNTAX	No. of bits	Mnemonics
ProgramInfo(){		
version_number	8*4	bslbf
length	32	uimsbf
if (length !=0){		
reserved	8	bslbf
number_of_program_sequences	8	uimsbf
for (i=0;i<*number_of_program_sequences*;i++){		
RSPN_program_sequence_start	32	uimsbf
reserved	48	bslbf
PCR_PID	16	bslbf
number_of_videos	8	uimsbf
number_of_audios	8	uimsbf
for (k=0;k<*number_of_videos*;k++){		
video_stream_PID	16	bslbf
VideoCodingInfo()		
}		
for (k=0;k<*number_of_audios*;k++){		
audio_stream_PID	16	bslbf
AudioCodingInfo()		
}		
}		
}		
}		

Syntax of ProgramInfo

[FIG.55]

SYNTAX	No. of bits	Mnemonics
VideoCodingInfo() {		
video_format	8	uimsbf
frame_rate	8	uimsbf
display_aspect_ratio	8	uimsbf
reserved	8	bslbf
}		

Syntax of VideoCondingInfo

[FIG.56]

video_format	MEANING
0	480i
1	576i
2	480p(including 640×480p format)
3	1080i
4	720p
5	1080p
6-254	reserved
255	No information

video_format

[FIG.57]

frame_rate	MEANING
0	forbidden
1	24 000/1001 (23.976...)
2	24
3	25
4	30 000/1001 (29.97..)
5	30
6	50
7	60 000/1001 (59.94..)
8	60
9-254	reserved
255	No information

frame_rate

[FIG.58]

display_aspect_ratio	MEANING
0	forbidden
1	reserved
2	4:3 display aspect ratio
3	16:9 display aspect ration
4-254	reserved
255	No information

display_aspect_ratio

[FIG.59]

SYNTAX	No. of bits	Mnemonics
AudioCodingInfo() {		
audio_format	8	uimsbf
audio_component_type	8	uimsbf
sampling_frequency	8	uimsbf
reserved	8	bslbf
}		

Syntax of AudioCondingInfo

[FIG.60]

audio_coding	MEANING
0	MPEG-1 audio layer I or II
1	Dolby AC-3 audio
2	MPEG-2 AAC
3	MPEG-2 multi-channel audio, backward compatible to MPEG-1
4	SESF LPCM audio
5-254	reserved
255	No information

audio_coding

[FIG.61]

audio_component_type	MEANING
0	single mono channel
1	dual mono channel
2	stereo (2-channel)
3	multi-lingual, multi-channel
4	surround sound
5	audio description for the visually impaired
6	audio for the hard of hearing
7-254	reserved
255	No information

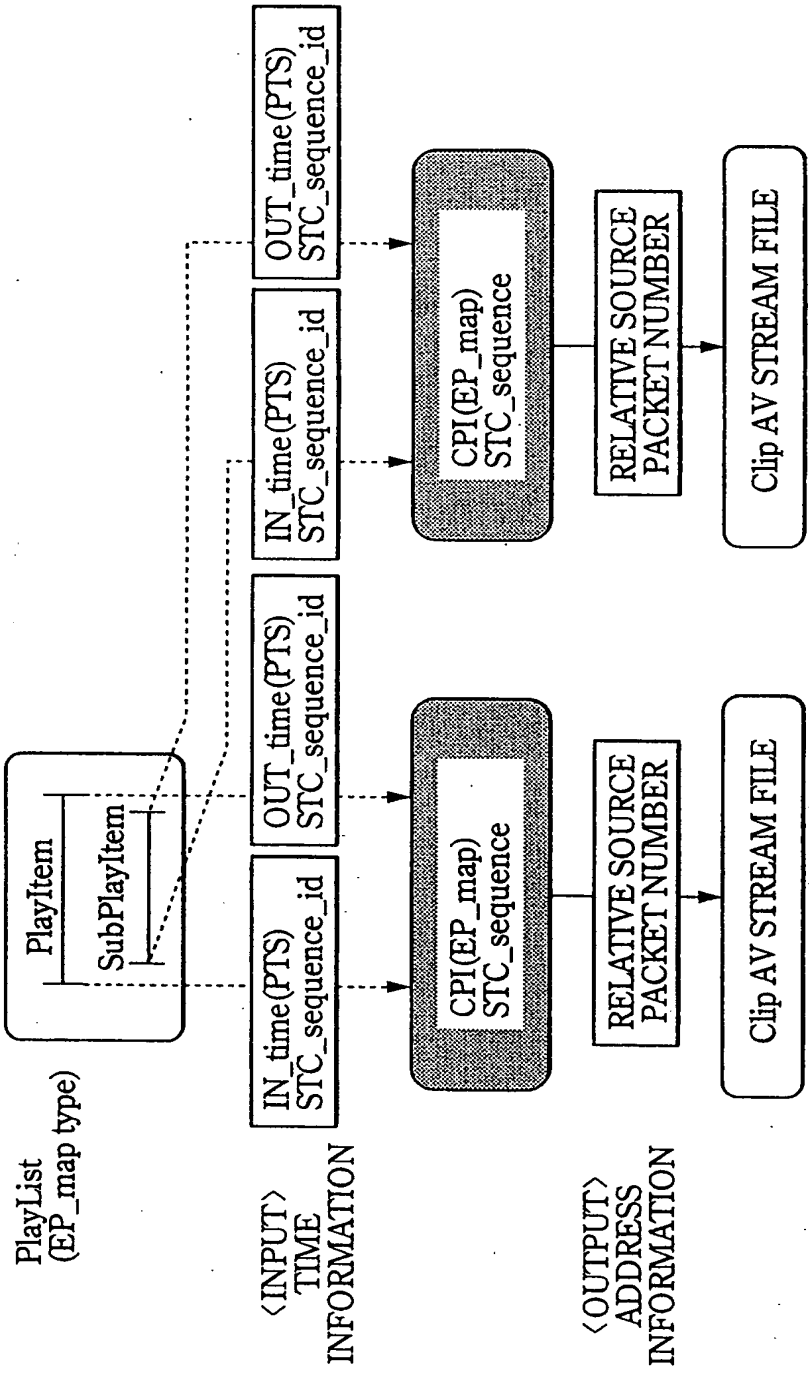
audio_component_type

[FIG.62]

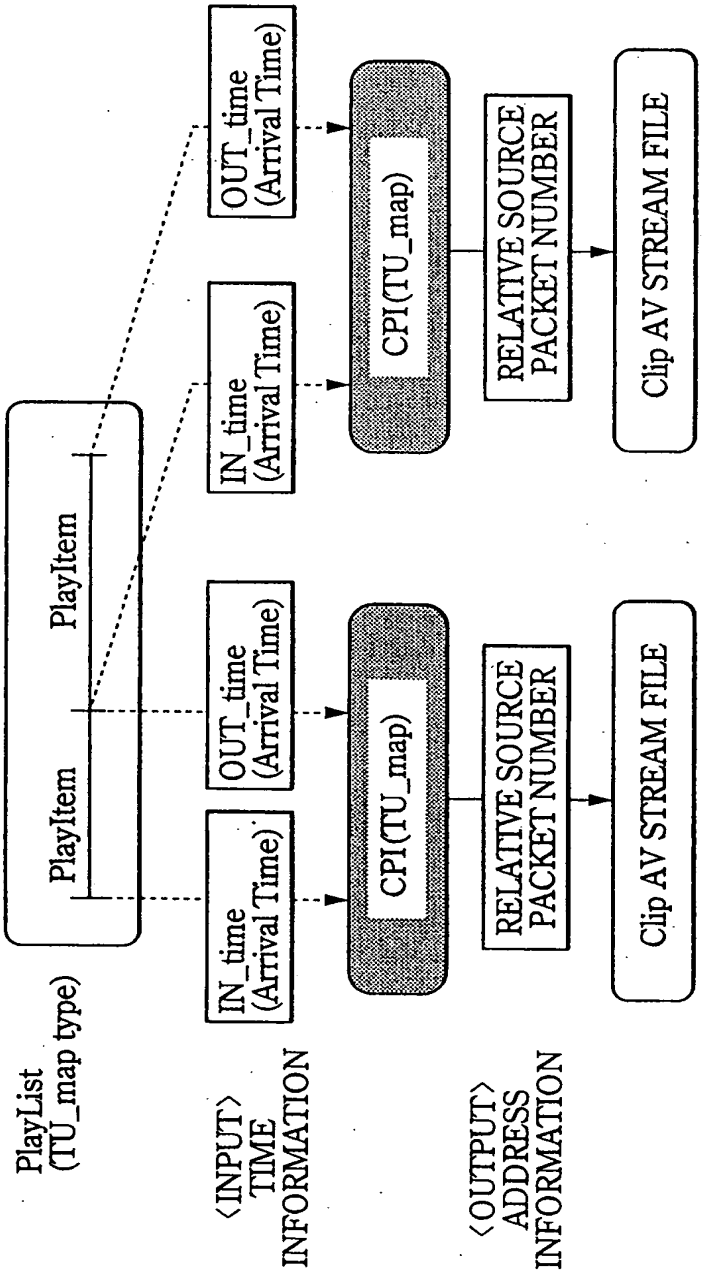
sampling_frequency	MEANING
0	48 kHz
1	44.1 kHz
2	32 kHz
3-254	reserved
255	No information

sampling_frequency

[FIG.63]


PlayList
(EP_map type)
PlayItem
SubPlayItem
〈INPUT〉
TIME
INFORMATION
IN_time(PTS)
STC_sequence_id
OUT_time(PTS)
STC_sequence_id
IN_time(PTS)
STC_sequence_id
OUT_time(PTS)
STC_sequence_id
〈OUTPUT〉
ADDRESS
INFORMATION
CPI(EP_map)
STC_sequence
CPI(EP_map)
STC_sequence
RELATIVE SOURCE
PACKET NUMBER
RELATIVE SOURCE
PACKET NUMBER
Clip AV STREAM FILE
Clip AV STREAM FILE

[FIG.64]


PlayList
(TU_map type)
PlayItem
PlayItem
<INPUT>
TIME
INFORMATION
IN_time
(Arrival Time)
OUT_time
(Arrival Time)
IN_time
(Arrival Time)
OUT_time
(Arrival Time)
<OUTPUT>
ADDRESS
INFORMATION
CPI(TU_map)
CPI(TU_map)
RELATIVE SOURCE
PACKET NUMBER
RELATIVE SOURCE
PACKET NUMBER
Clip AV STREAM FILE
Clip AV STREAM FILE

[FIG.65]

SYNTAX	No. of bits	Mnemonics
CPI(){		
version_number	8*4	bslbf
length	32	uimsbf
reserved	15	bslbf
CPI_type	1	bslbf
if (CPI_type==0)		
EP_map()		
else		
TU_map()		
}		

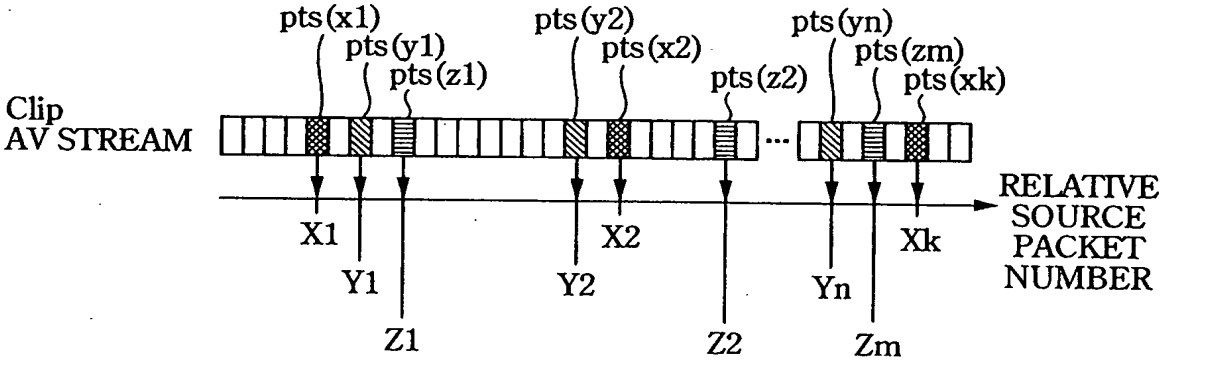
Syntax of CPI

[FIG.66]

CPI_type	MEANING
0	EP map type
1	TU map type

CPI_type

[FIG.67]



: SOURCE PACKET CONTAINING FIRST BYTE OF SEQUENCE HEADER video_PID=x

: SOURCE PACKET CONTAINING FIRST BYTE OF SEQUENCE HEADER video_PID=y

: SOURCE PACKET CONTAINING FIRST BYTE OF SEQUENCE HEADER video_PID=z

EP_map

number_of_stream/PIDs=3

EP_map_for_ one_stream_PID(0)

stream_PID(0)=x
num_EP_entries(0)=k

PTS_EP start	RSPN_EP start
pts(x1)	X1
pts(x2)	X2
...	...
pts(xk)	Xk

EP_map_for_ one_stream_PID(1)

stream_PID(1)=y
num_EP_entries(1)=n

PTS_EP start	RSPN_EP start
pts(y1)	Y1
pts(y2)	Y2
...	...
pts(yn)	Yn

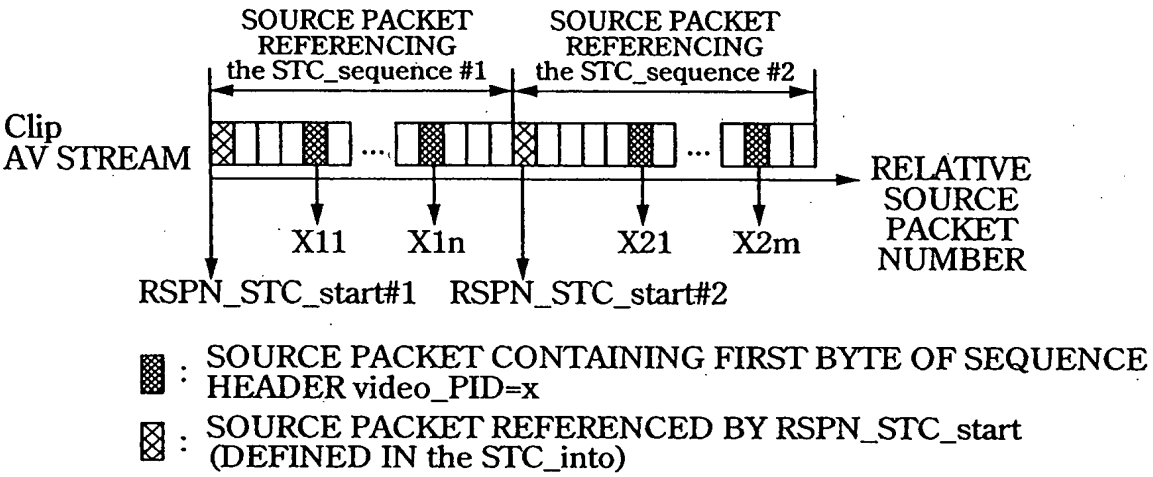
EP_map_for_ one_stream_PID(2)

stream_PID(2)=z
num_EP_entries(2)=m

PTS_EP start	RSPN_EP start
pts(z1)	Z1
pts(z2)	Z2
...	...
pts(zm)	Zm

Video EP_map

[FIG.68]



EP_map_for_one_steram_ PID
video_PID=x

PTS_EP start	RSPN_EP start
pts(x11)	X11
...	...
pts(x1n)	X1n
pts(x21)	X21
...	...
pts(x2m)	X2m

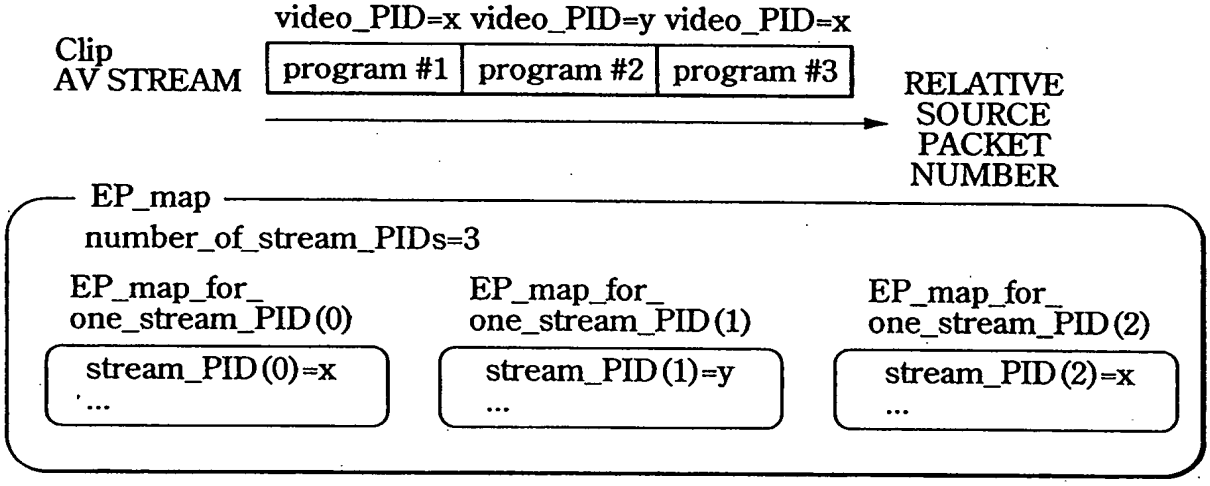
DATA BELONGING TO STC_sequence #1

boundary

DATA BELONGING TO STC_sequence #2

RSPN_STC_start #2<X21

[FIG.69]



[FIG.70]

SYNTAX	No. of bits	Mnemonics
Thumbnail(){		
version_number	8*4	char
length	32	uimsbf
if (length !=0){		
tn_blocks_start_address	32	bslbf
number_of_thumbnails	16	uimsbf
tn_block_size	16	uimsbf
number_of_tn_blocks	16	uimsbf
reserved	16	bslbf
for (i=0; i<*number_of_thumbnails*; i++){		
thumbnail_index	16	uimsbf
thumbnail_picture_format	8	bslbf
reserved	8	bslbf
picture_data_size	32	uimsbf
start_tn_block_number	16	uimsbf
x_picture_length	16	uimsbf
y_picture_length	16	uimsbf
reserved	16	uimsbf
}		
stuffing_bytes	8*2*L1	bslbf
for(k=0; k<*number_of_tn_blocks*; k++){		
tn_block	tn_block_ size*1024*8	
}		
}		
}		

Syntax of Thumbnail

[FIG.71]

EP_type	MEANING
0	video
1	audio
2-15	reserved

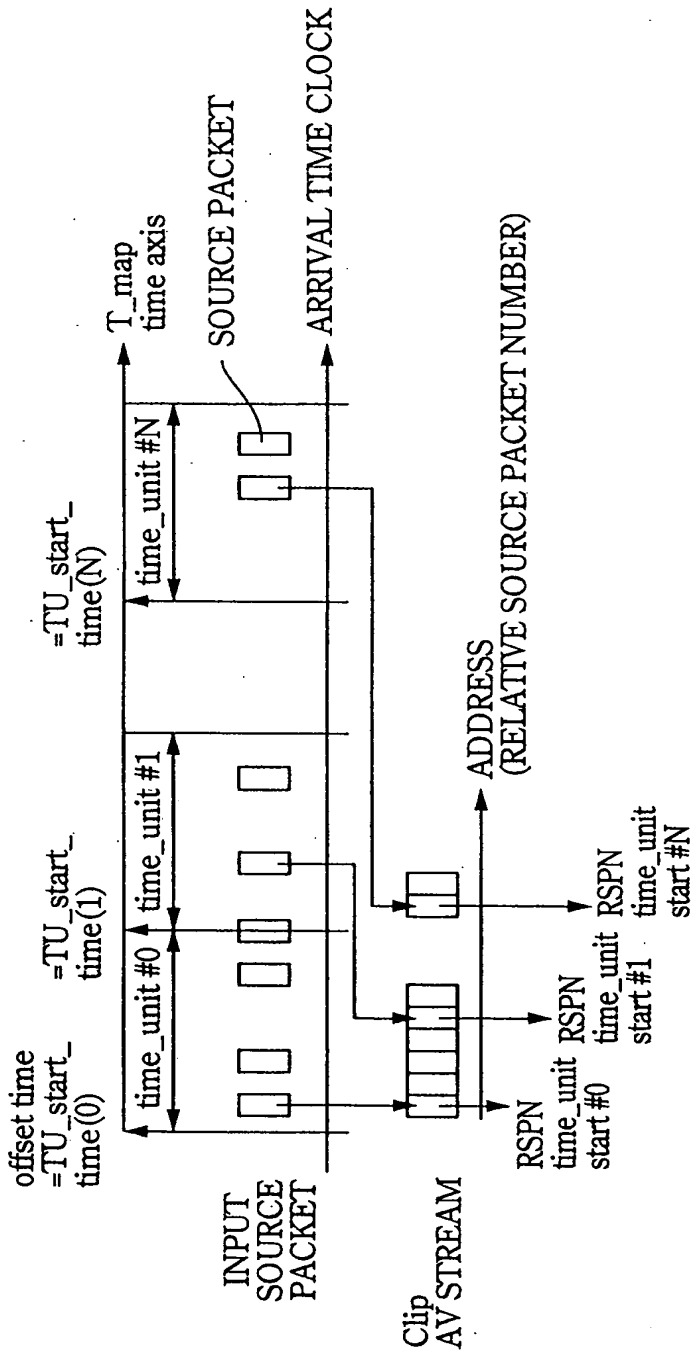
EP_typevalues

[FIG.72]

SYNTAX	No. of bits	Mnemonics
EP_map_for_one_stream_PID(*N*) {		
for (i=0;i<*N*;i++){		
PTS_EP_start	32	uimsbf
RSPN_EP_start	32	uimsbf
}		
}		

Syntax of EP_map_for_one_stream_PID

[FIG.73]


offset time
=TU_start_
time(0)
=TU_start_
time(1)
=TU_start_
time(N)
time_unit #0
time_unit #1
time_unit #N
T_map
time axis
SOURCE PACKET
ARRIVAL TIME CLOCK
INPUT
SOURCE
PACKET
Clip
AV STREAM
ADDRESS
(RELATIVE SOURCE PACKET NUMBER)
RSPN
time_unit
start #0
RSPN
time_unit
start #1
RSPN
time_unit
start #N

[FIG.74]

SYNTAX	No. of bits	Mnemonics
TU_map() {		
offset_time	32	bslbf
time_unit_size	32	uimsbf
number_of_time_unit_entries	32	uimsbf
for (k=0;k<*number_of_time_unit_entries*;k++)		
RSPN_time_unit_start	32	uimsbf
}		

Syntax of TU_map

[FIG.75]

SYNTAX	No. of bits	Mnemonics
ClipMark(){		
version_number	8*4	bslbf
length	32	uimsbf
number_of_Clip_marks	16	uimsbf
for (i=0; i<*number_of_clip_marks*; i++){		
reserved	8	bslbf
mark_type	8	bslbf
mark_time_stamp	32	uimsbf
STC_sequence_id	8	uimsbf
reserved	24	bslbf
character_set	8	bslbf
name_length	8	uimsbf
mark_name	8*256	bslbf
ref_thumbnail_index	16	uimsbf
}		
}		

Syntax of ClipMark

[FIG.76]

Mark_type	MEANING	COMMENT
0x00-0x8F	reserved	Reserved for PlayListMark0
0x90	Event-start mark	MARK POINT INDICATING PROGRAM START POINT
0x91	Local event-start mark	MARK POINT INDICATING LOCAL SCENE IN PROGRAM
0x92	Scene-start mark	MARK POINT SHOWING SCENE CHANGE POINT
0x93-0xFF	reserved	

mark_type

[FIG.77]

CPI_type in the PlayList()	SEMANTICS OF mark_time_stamp
EP_map type	mark_time_stamp MUST INDICATE UPPER 32 BITS OF 33 BIT LENGTH PTS CORRESPONDING TO PRESENTATION UNIT REFERENCED BY MARK.
TU_map type	mark_time_stamp MUST BE TIME ON TU_map_time_axis AND MUST BE ROUNDED TO time_unit PRECISION. mark_time_stamp IS CALCULATED BY FOLLOWING EQUATION: $mark\_time\_stamp = TU\_start\_time \ \%2^{32}$

mark_type_stamp

[FIG.78]

SYNTAX	No. of bits	Mnemonics
menu.thmb/mark.thmb() {		
reserved	256	bslbf
Thumbnail()		
for (i=0;i<N1;i++)		
padding_word	16	bslbf
}		

Syntax of menu.thmb and mark.thmb

[FIG.79]

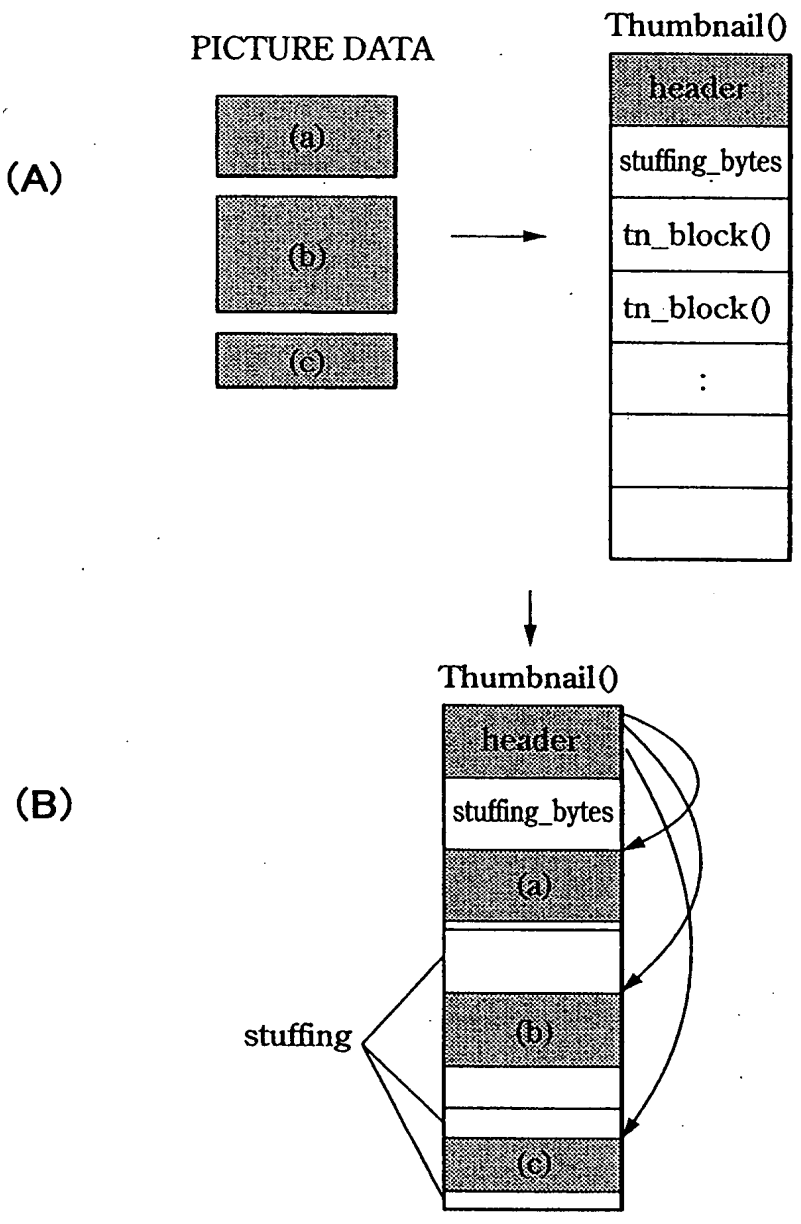
SYNTAX	No. of bits	Mnemonics
Thumbnail(){		
version_number	8*4	char
length	32	uimsbf
if (length !=0){		
tn_blocks_start_address	32	bslbf
number_of_thumbnails	16	uimsbf
tn_block_size	16	uimsbf
number_of_tn_blocks	16	uimsbf
reserved	16	bslbf
for (i=0; i<*number_of_thumbnails*; i++){		
thumbnail_index	16	uimsbf
thumbnail_picture_format	8	bslbf
reserved	8	bslbf
picture_data_size	32	uimsbf
start_tn_block_number	16	uimsbf
x_picture_length	16	uimsbf
y_picture_length	16	uimsbf
reserved	16	uimsbf
}		
stuffing_bytes	8*2*L1	bslbf
for(k=0; k<*number_of_tn_blocks*; k++){		
tn_block	tn_block_ size*1024*8	
}		
}		
}		

Syntax of Thumbnail

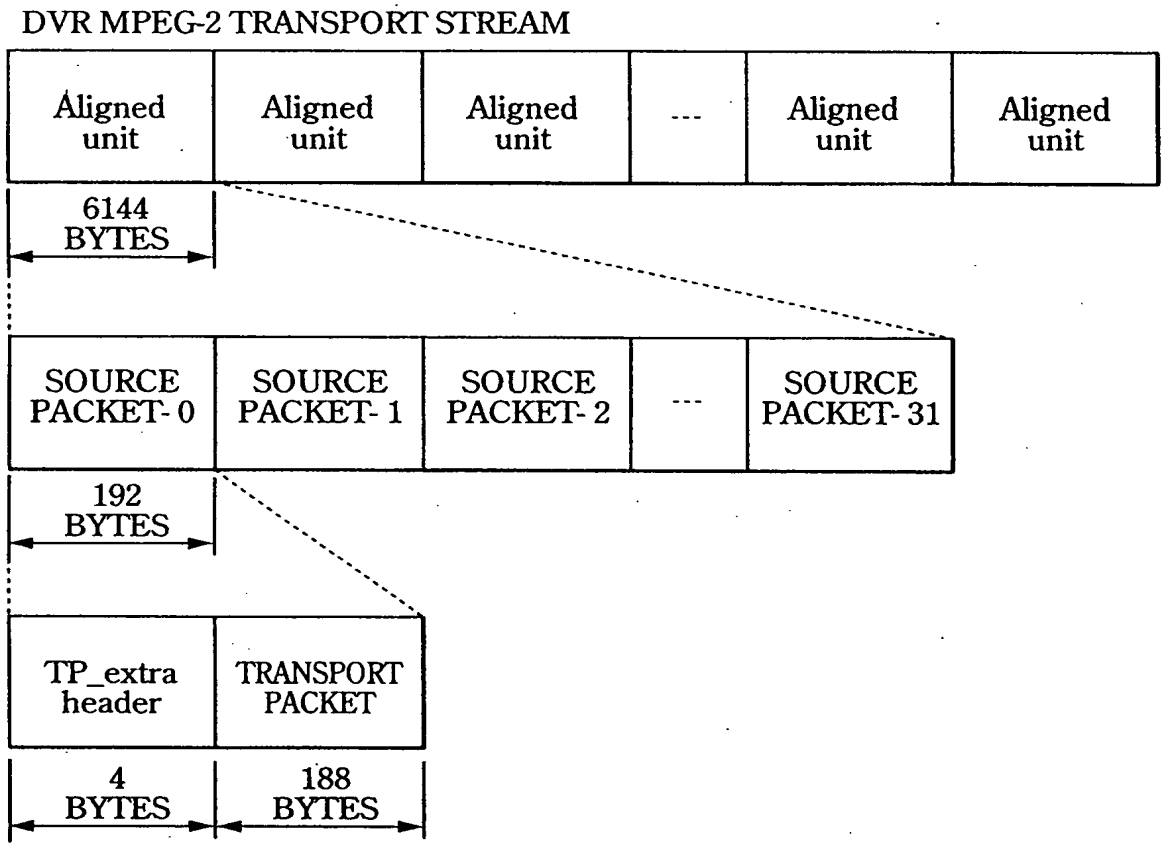
[FIG.80]

Thumbnail_picture_format	MEANING
0x00	MPEG-2 Video I-picture
0x01	DCF (restricted JPEG)
0x02	PNG
0x03-0xff	reserved

[FIG.81]

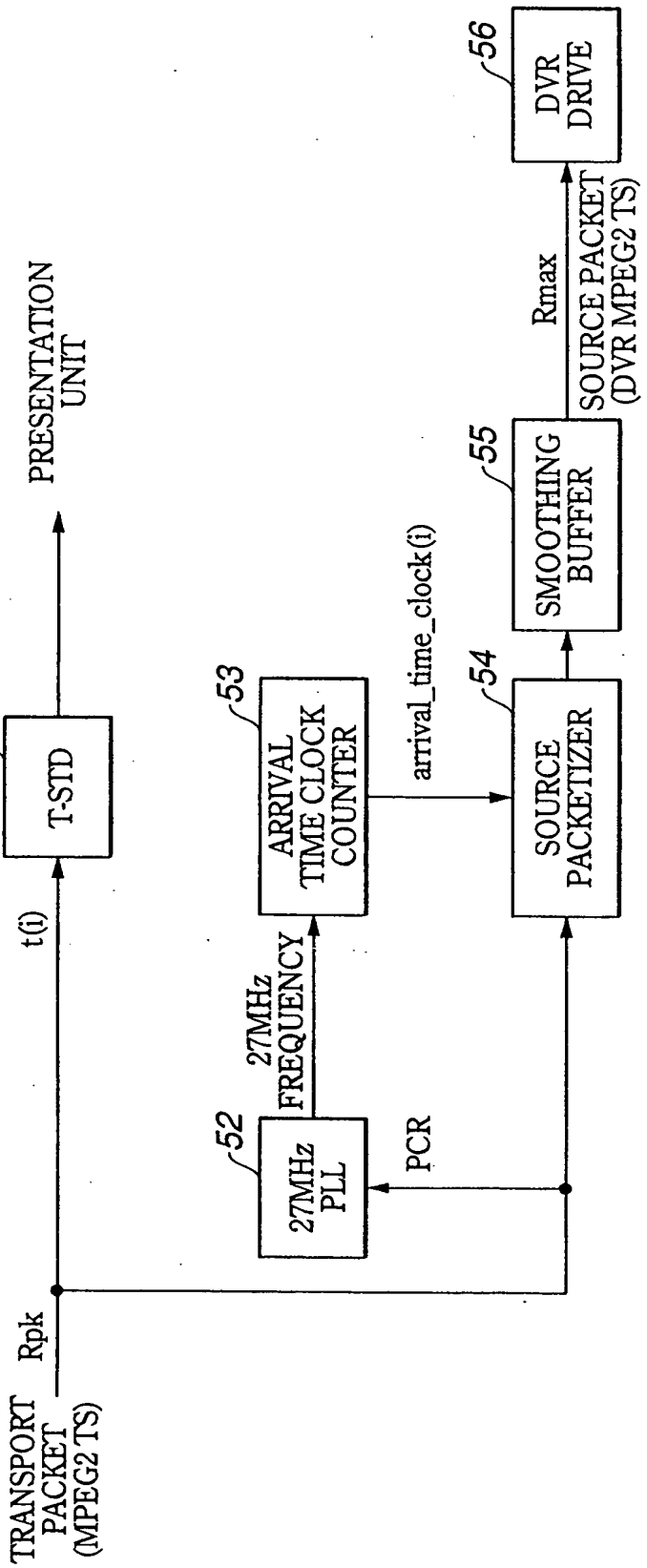


[FIG.82]



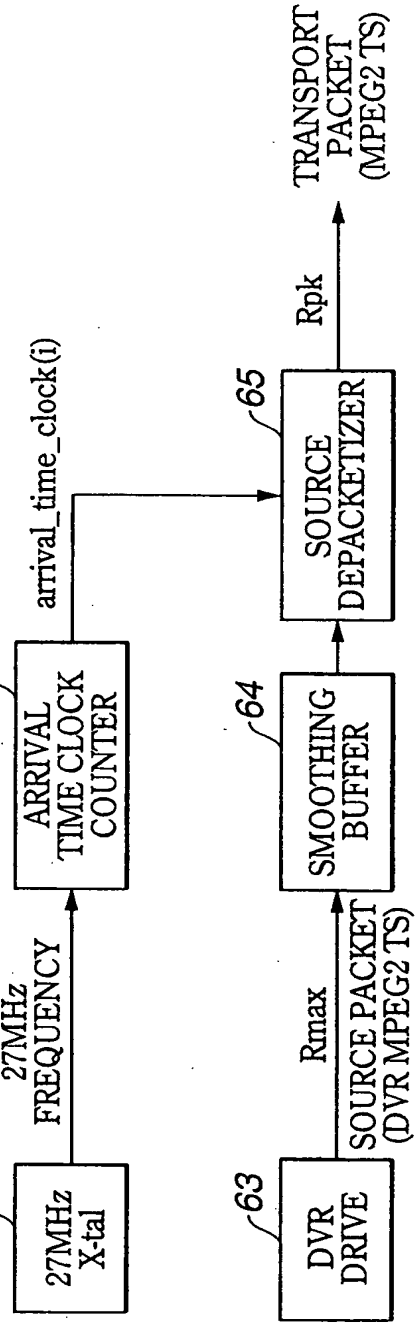
Structure of transport stream of DVR MPEG2

[FIG.83]



Recorder model of transport stream of DVR MPEG2

[FIG.84]



Player model of transport stream of DVR MPEG2

[FIG.85]

SYNTAX	No. of bits	Mnemonics
source_packet() {		
TP_extra_header()		
trasport_packet()		
}		

source packet

[FIG.86]

SYNTAX	No. of bits	Mnemonics
TP_extra_header() {		
copy_permission_indicator	2	uimsbf
arrival_time_stamp	30	uimsbf
}		

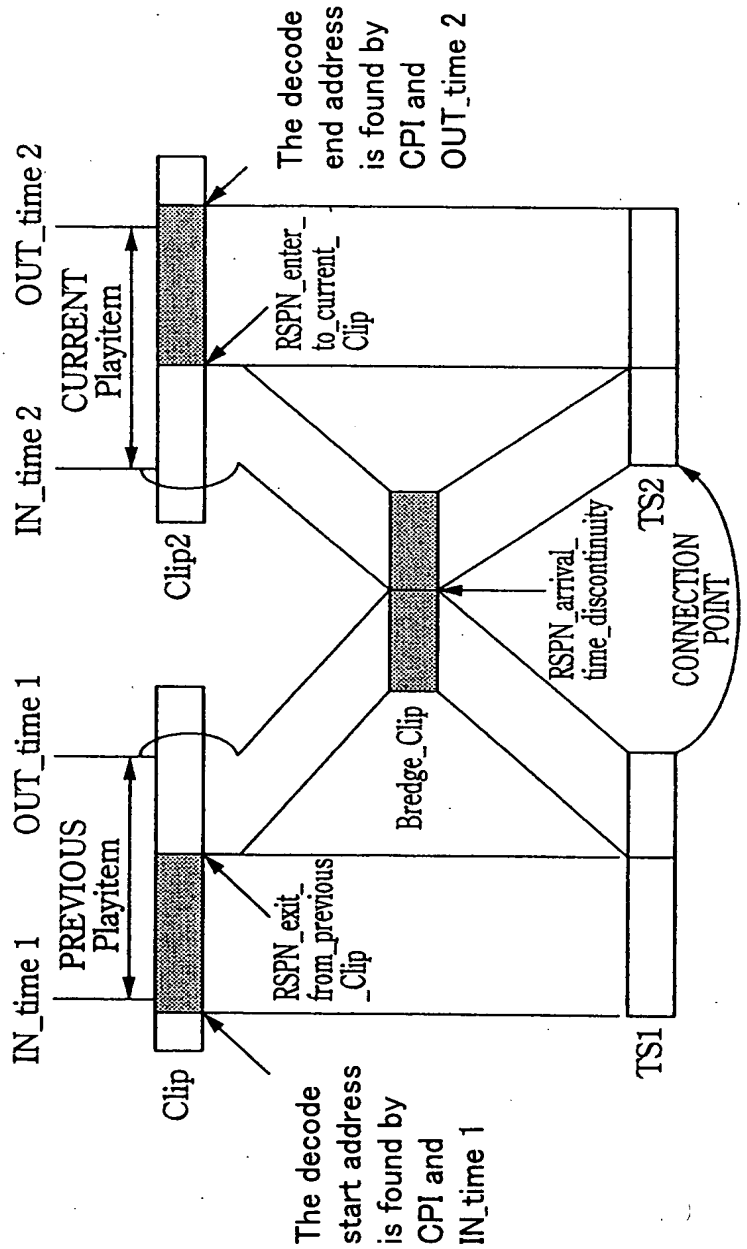
TP_extra_header

[FIG.87]

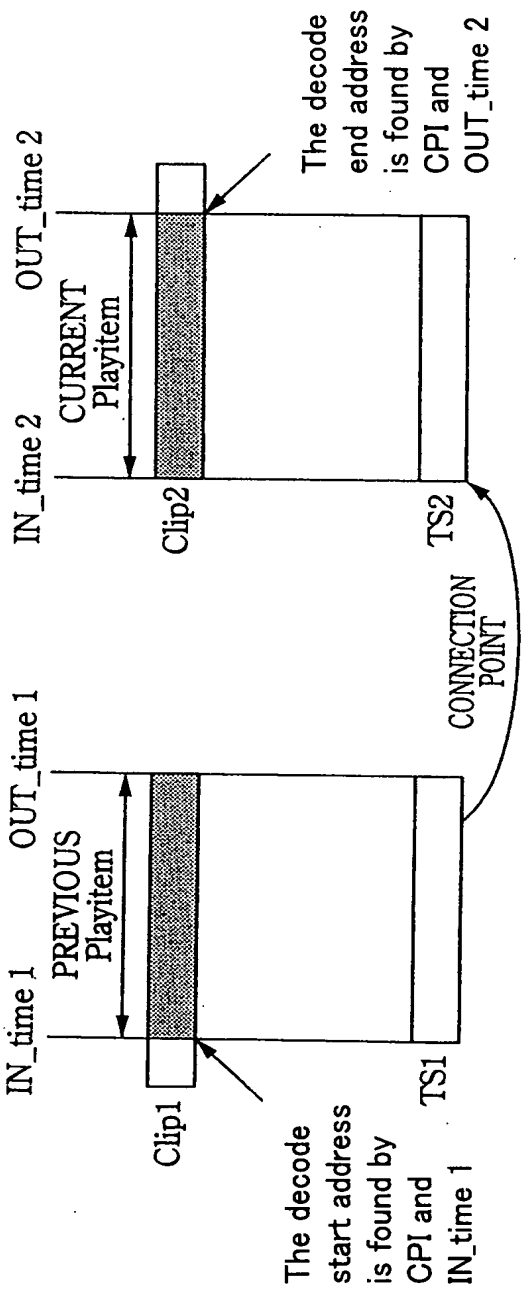
copy_permission _indicator	MEANING
00	copy free
01	no more copy
10	copy once
11	copy prohibited

copy permission indicator table

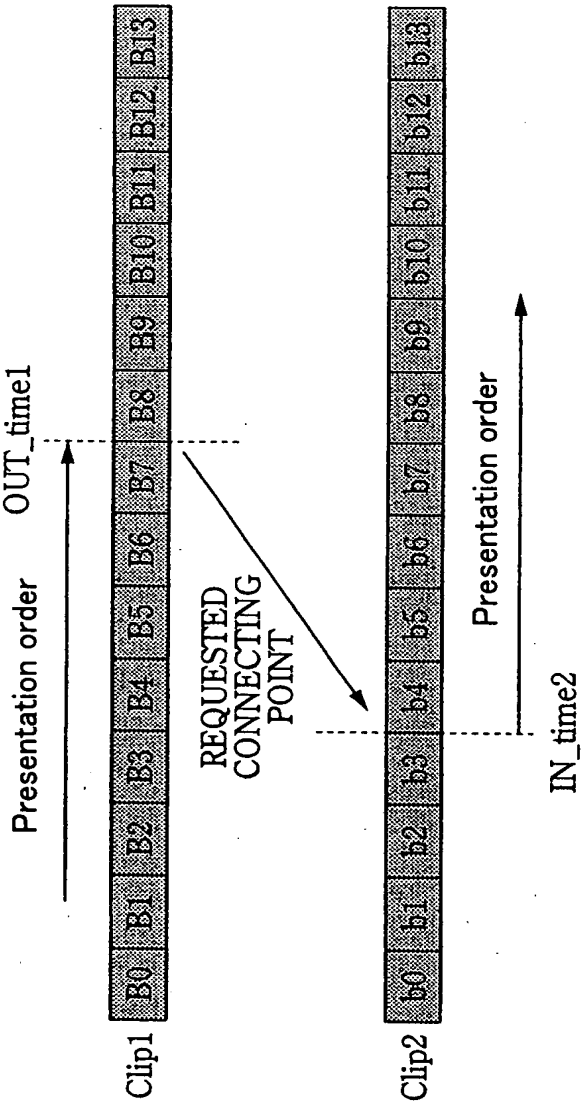
[FIG.88]


IN_time 1
OUT_time 1
PREVIOUS Playitem
IN_time 2
OUT_time 2
CURRENT Playitem
Clip
Clip2
Bredge_Clip
RSPN_exit_from_previous_Clip
RSPN_enter_to_current_Clip
RSPN_arrival_time_discontinuity
TS1
TS2
CONNECTION POINT
The decode start address is found by CPI and IN_time 1
The decode end address is found by CPI and OUT_time 2

[FIG.89]

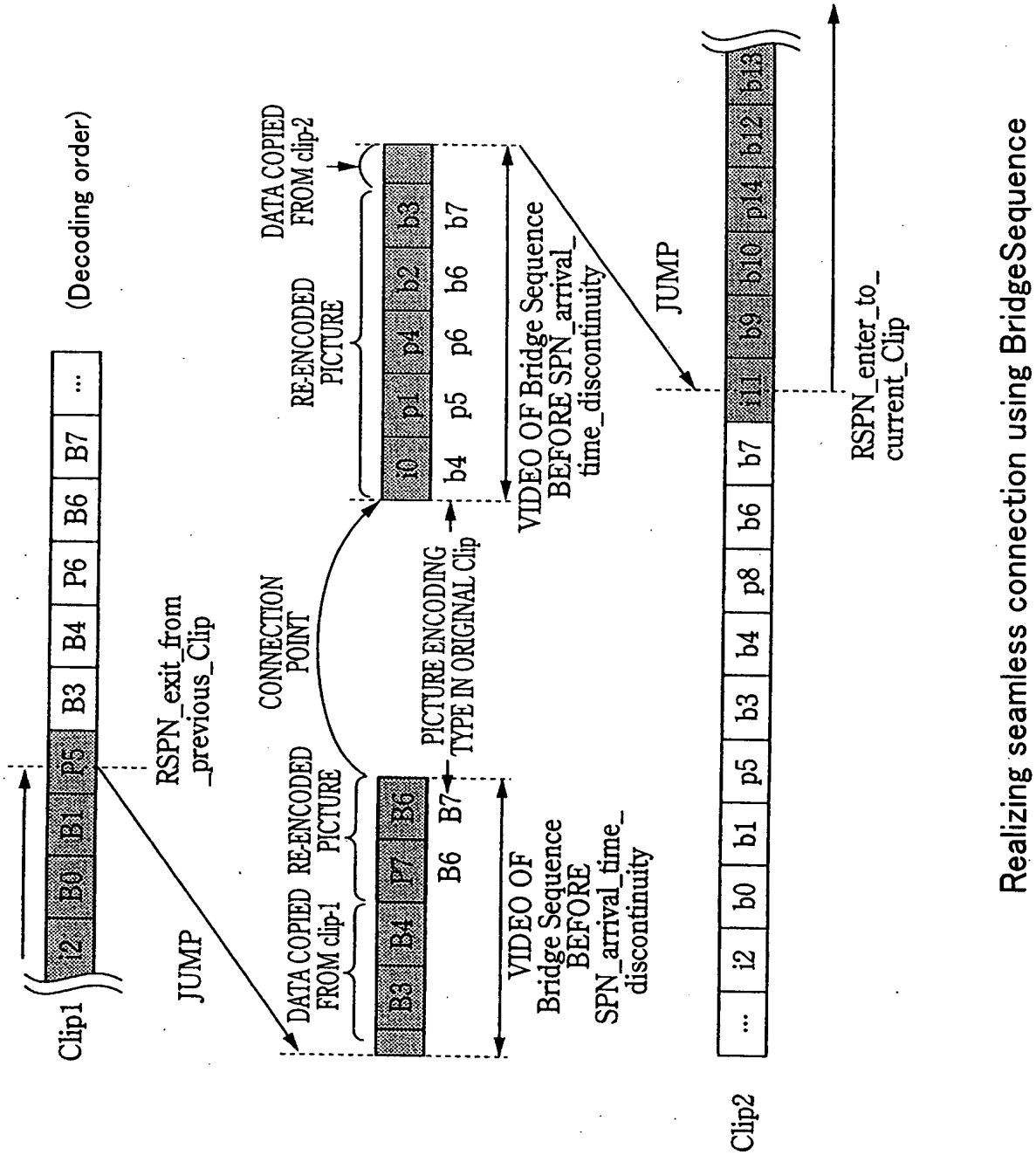

IN_time 1
OUT_time 1
PREVIOUS Playitem
Clip1
TS1
The decode start address is found by CPI and IN_time 1
CONNECTION POINT
IN_time 2
OUT_time 2
CURRENT Playitem
Clip2
TS2
The decode end address is found by CPI and OUT_time 2

[FIG.90]



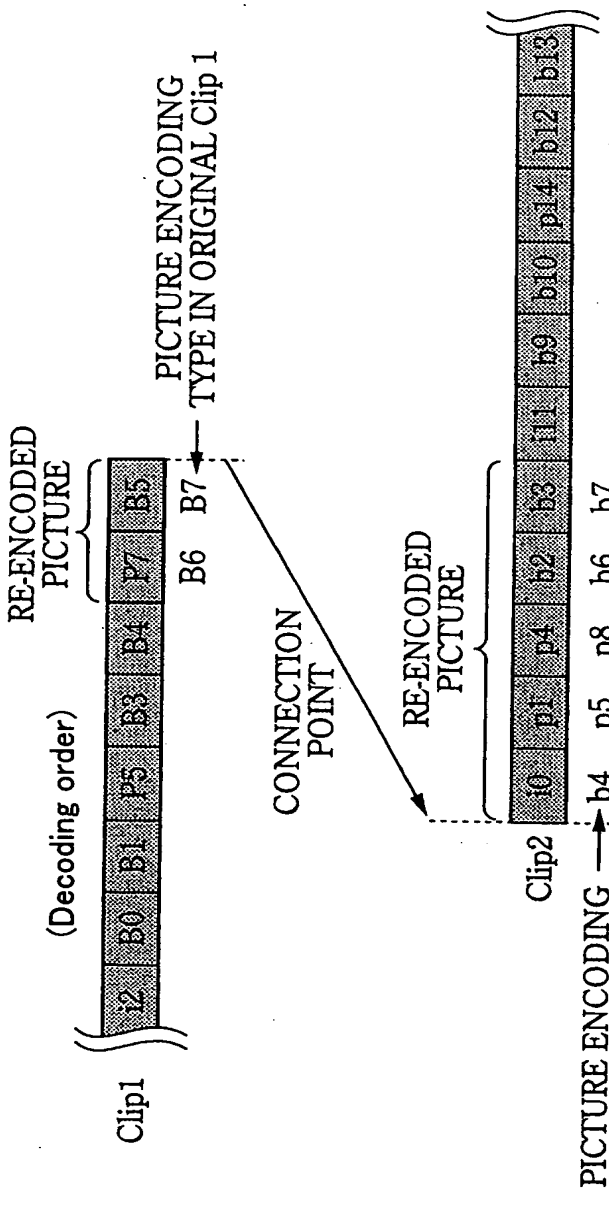
Seamless connection shown in picture display sequence

[FIG.91]



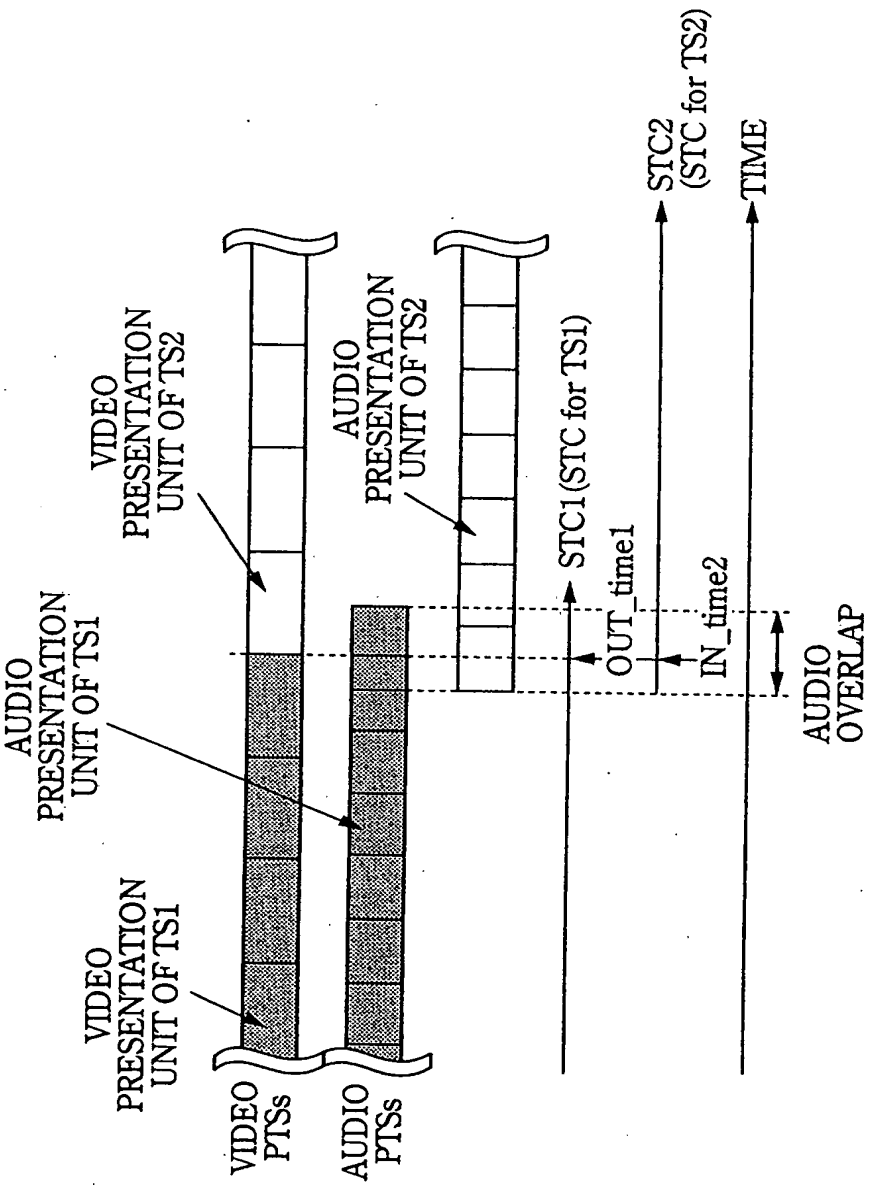
Realizing seamless connection using BridgeSequence

[FIG.92]

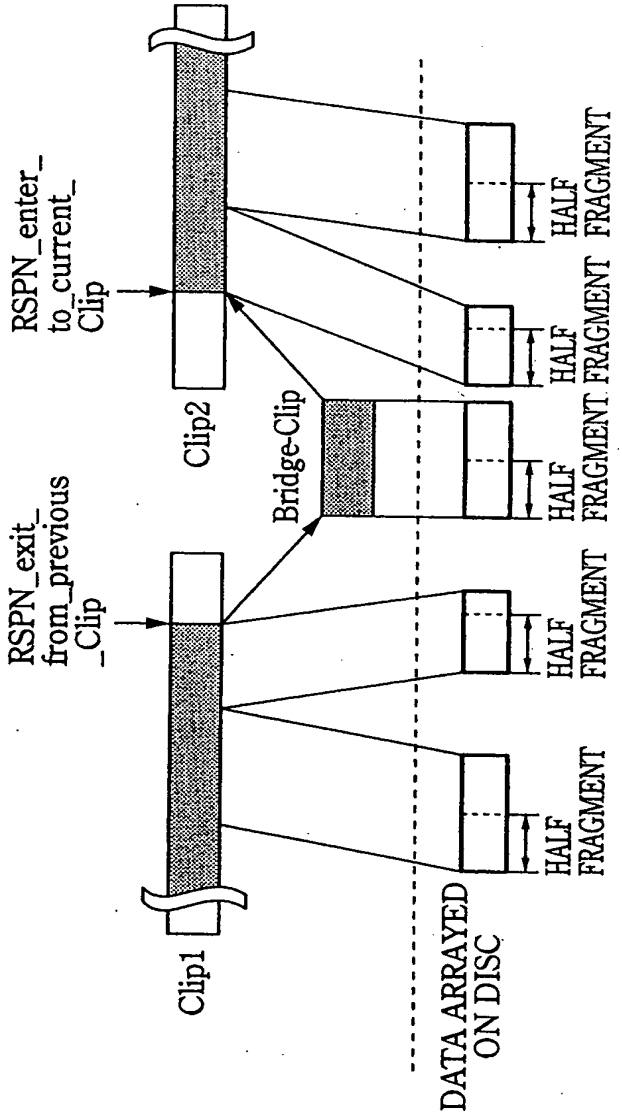


Realizing seamless connection without using BridgeSequence

[FIG.93]

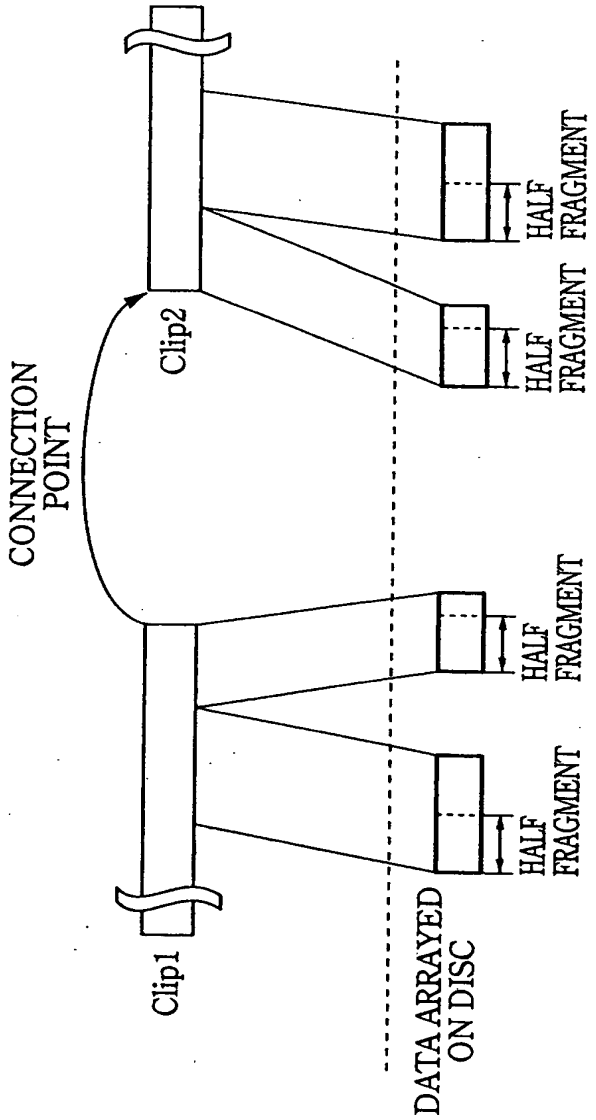

VIDEO PRESENTATION UNIT OF TS1
AUDIO PRESENTATION UNIT OF TS1
VIDEO PRESENTATION UNIT OF TS2
AUDIO PRESENTATION UNIT OF TS2
VIDEO PTSs
AUDIO PTSs
STC1 (STC for TS1)
STC2 (STC for TS2)
TIME
OUT_time1
IN_time2
AUDIO OVERLAP

[FIG.94]



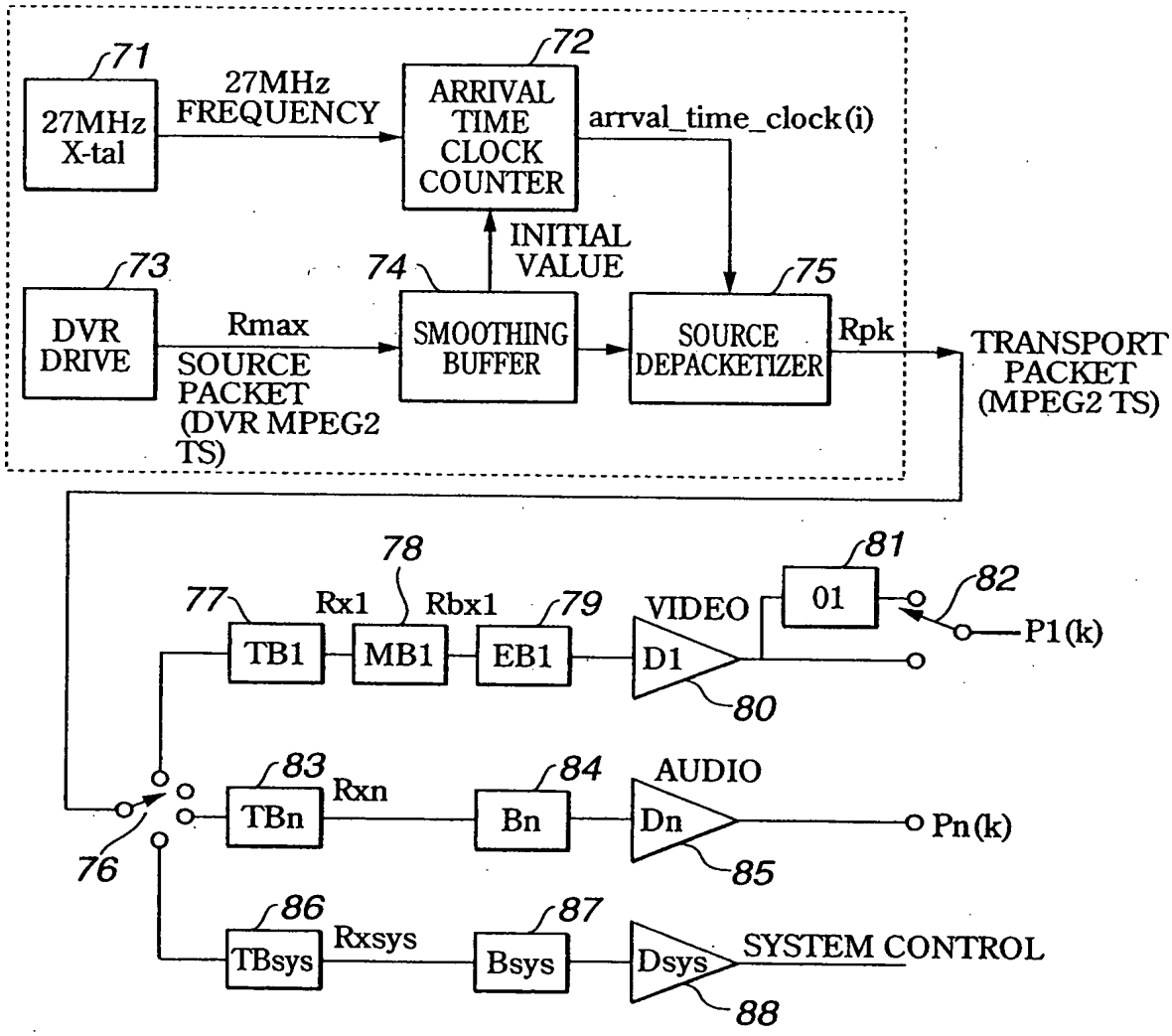
Data allocation in case of creating seamless connection using BridgeSequence

[FIG.95]

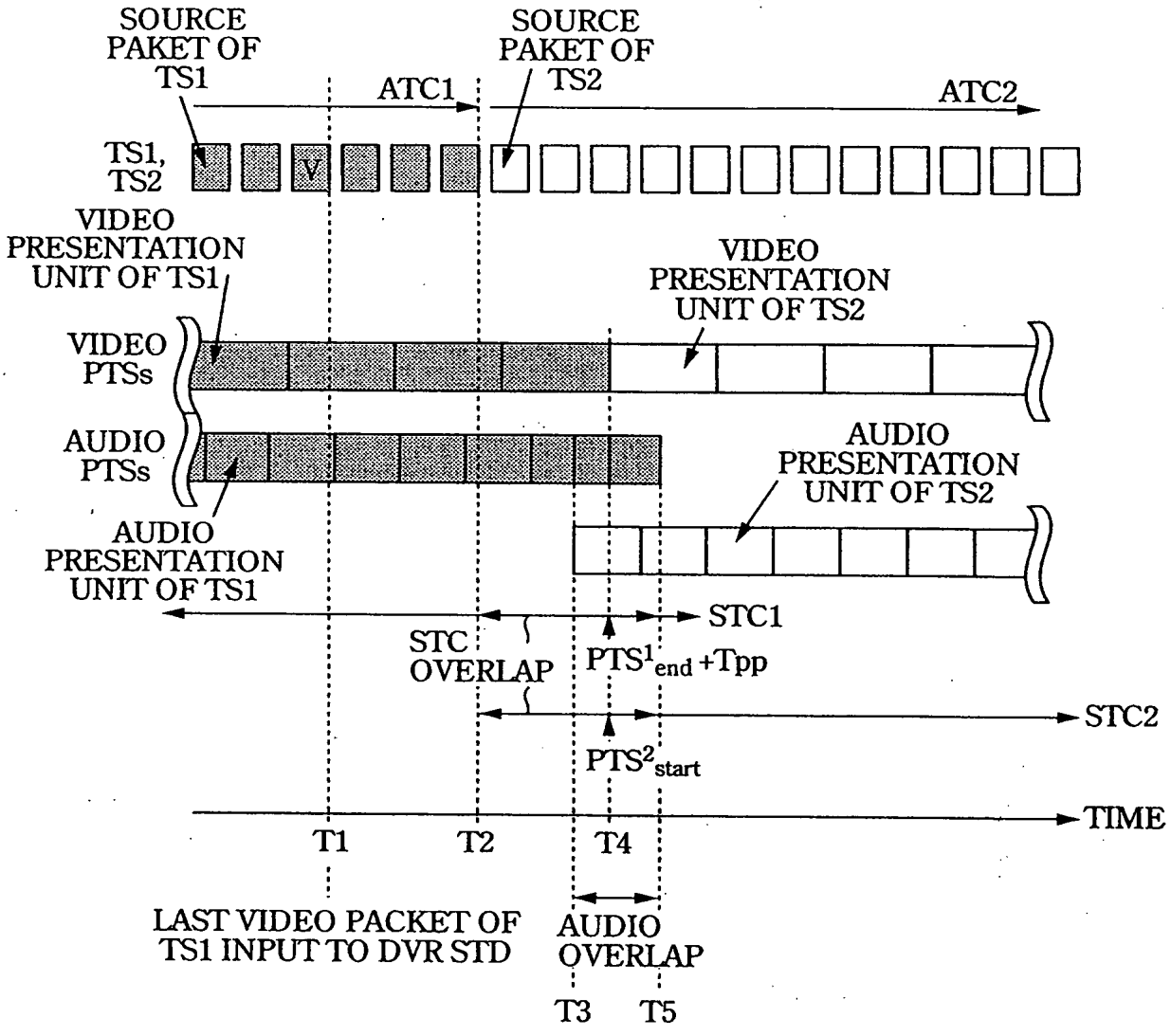


Data allocation in case of creating seamless connection without using BridgeSequence

[FIG.96]

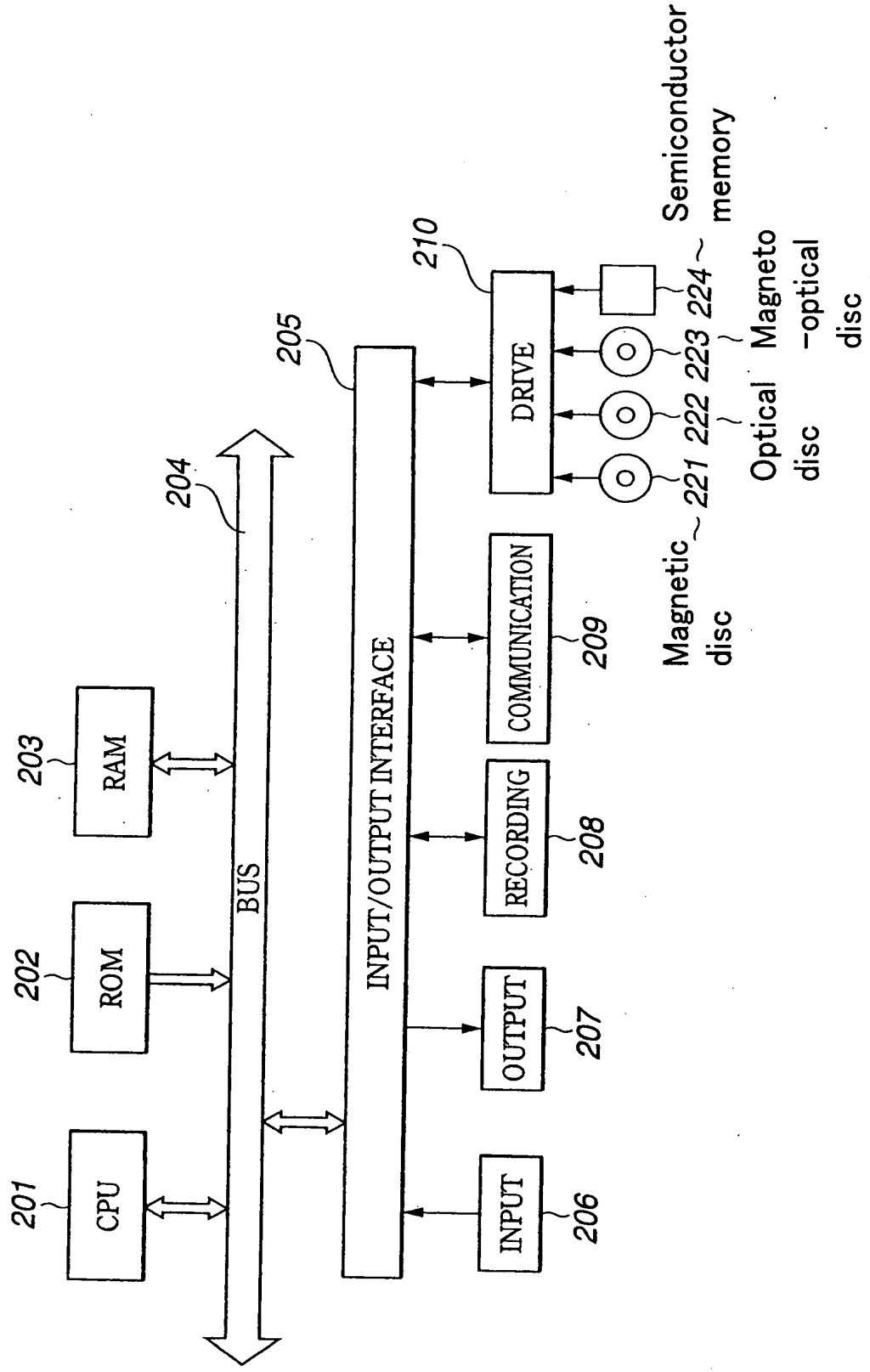

71
27MHz X-tal
27MHz FREQUENCY
72
ARRIVAL TIME CLOCK COUNTER
arrval_time_clock(i)
INITIAL VALUE
73
DVR DRIVE
Rmax
SOURCE PACKET (DVR MPEG2 TS)
74
SMOOTHING BUFFER
75
SOURCE DEPACKETIZER
Rpk
TRANSPORT PACKET (MPEG2 TS)
78
77
Rx1
Rbx1
79
VIDEO
81
01
82
P1(k)
TB1
MB1
EB1
D1
80
83
Rxn
84
AUDIO
TBn
Bn
Dn
Pn(k)
85
76
86
Rxsys
87
SYSTEM CONTROL
TBsys
Bsys
Dsys
88

[FIG.97]



Timing diagram for inputting, decoding and displaying transport packet when transferring from given AV stream (TS1) to the next AV stream (TS2) seamlessly connected thereto

[FIG.98]


201
CPU
202
ROM
203
RAM
204
BUS
205
INPUT/OUTPUT INTERFACE
206
INPUT
207
OUTPUT
208
RECORDING
209
COMMUNICATION
210
DRIVE
221
Magnetic disc
222
Optical disc
223
Magneto -optical disc
224
Semiconductor memory

[Name of Document] ABSTRACT

[Summary]

[Task]

Determinationn of readout position and/or decode processing of AV stream are permitted to be quickly performed.

[Means for solution]

For every ClipInfo, attribute information of AV stream are stored. The Offset_SPN in ClipInfo gives offset value of source packet No. with respect to the first source packet of AV stream. The time_controlled_flag indicates whether or not corresponding mode is recording mode where AV stream is recorded in such a manner that file size becomes proportional with respect to time elapse. Other data are data indicating type of AV stream, and/or data relating to date recorded. By referring these data, it is possible to perform determination of readout position of AV stream.

[Selected Drawing] Fig. 46